Attachment G3 - Technical Management Services and Deliverables

Part 1, Technical Management Approach (TMA)

Part 2, Technical Management Statement of Work (SOWs)

Part 3, Technical Management Contract Deliverable List Item Descriptions

The following change orders have revised this Rider:

AMEND #	CHANGE ORDER NUMBER
1	None
2	CR-00007, CR-00017, CR-00023, CR-00025, CR-00030, CR-00033, CR-00044,
3	CR-00042, CR-00049, CR-00050, CR-00053, CR-00056, CR-00057, CR-00061, CR-00064, CR-00069,
4	CR-00073, CR-00075, CR-00083-1, CR-00083-2, CR-0083-3, CR-00087, CR-2-00108-01, CR-2-00113-01, CR-2-00119,
5	CR-2-00124a, CR-2-00142-01,
6	None
7	CR-2-00233, CR-2-00247,
8	None
9	CR-2-00295, CR-2-00296,
10	CR-2-01016, None
11	CR-2-01020, CR-2-01072,
12	CR-2-01124, CR-C-01157, and CR-C-01167

TECHNICAL MANAGEMENT APPROACH

3C: Technical Management Approach (TMA DID)

CCSAS
Document Number SCP - FTB - 00 -1730 –150
Volume 1, FINAL

Document Review and Change Summary			
Version Number	**Issue Date**	**Author(s)**	**Brief Summary of Changes from Prior Version**

TABLE OF CONTENTS

3C: TECHNICAL MANAGEMENT APPROACH (TMA DID) 11
3C.1 INTRODUCTION 11
3C.1.1 PURPOSE 11
3C.1.2 DOCUMENT OVERVIEW 11
3C.1.3 DEFINITIONS 12
3C.2 REFERENCED DOCUMENTS 13
3C.2.1 INTRODUCTION 13
3C.2.2 REFERENCED DOCUMENTS LIST 13
3C.3 OVERVIEW OF TECHNICAL MANAGEMENT APPROACH 14
3C.3.1 INTRODUCTION 14
3C.3.2 PROACTIVE, NOT REACTIVE 15
3C.3.3 CONCLUSION 23
3C.4 CONFIGURATION MANAGEMENT 24
3C.4.1 INTRODUCTION 24
3C.4.2 OVERVIEW AND OBJECTIVES 24
3C.4.3 CONFIGURATION MANAGEMENT ENVIRONMENTS 29
3C.4.4 CONFIGURATION BASELINE MANAGEMENT 32
3C.4.5 DATABASE ITEMS 34
3C.4.6 TECHNICAL DOCUMENTATION 35
3C.4.7 CONCLUSION 37
3C.5 REQUIREMENTS MANAGEMENT 38
3C.5.1 INTRODUCTION 38
3C.5.2 OVERVIEW OF REQUIREMENT MANAGEMENT APPROACH 39
3C.5.3 MANAGING CHANGES PRIOR TO STATE ACCEPTANCE 41
3C.5.4 DOCUMENTING, MAINTAINING, MEASURING, REPORTING, AND CONTROLLING CHANGES 42
3C.5.5 ASSESSING IMPACT OF REQUIREMENTS CHANGES 45
3C.5.6 REQUIREMENT TRACEABILITY INTO LIFE CYCLE PRODUCTS 46
3C.5.7 CONCLUSION 49

3C.6 TECHNICAL RISK MANAGEMENT 51
3C.6.1 INTRODUCTION 51
3C.6.2 TECHNICAL RISK MANAGEMENT APPROACH 51
3C.6.3 TECHNICAL RISK IDENTIFICATION, QUANTIFICATION, AND MITIGATION 54
3C.6.4 CONCLUSION 65
3C.7 KNOWLEDGE MANAGEMENT 66
3C.7.1 INTRODUCTION 66
3C.7.2 CHALLENGES 67
3C.7.3 STRUCTURED KNOWLEDGE MANAGEMENT APPROACH 70
3C.7.4 PRODUCTION SUPPORT AND PROJECT TURN OVER 78
3C.7.5 KNOWLEDGE MANAGEMENT CURRICULUM 79
3C.7.6 CONCLUSION 82
3C.8 TECHNICAL REVIEWS 83
3C.8.1 INTRODUCTION 83
3C.8.2 FORMAL TECHNICAL REVIEWS 84
3C.8.3 INFORMAL TECHNICAL REVIEWS 86
3C.8.4 TECHNICAL REVIEW SCHEDULE 88
3C.8.5 TECHNICAL REVIEW APPROACH 88
3C.8.6 CONCLUSION 90
3C.9 TECHNICAL QUALITY MANAGEMENT 91
3C.9.1 INTRODUCTION 91
3C.9.2 EXPECTING VS. INSPECTING QUALITY 91
3C.9.3 TECHNICAL QUALITY ACTIVITIES THROUGHOUT THE DEVELOPMENT CYCLE 93
3C.9.4 RELATIONSHIP TO THE QUALITY MANAGEMENT PROGRAM 96
3C.9.5 ENFORCING INDEPENDENCE 96
3C.9.6 TECHNICAL QUALITY METRICS 99
3C.9.7 CONCLUSION 100
3C.10 PROBLEM RESOLUTION 101
3C.10.1 INTRODUCTION 101
3C.10.2 PROBLEM RESOLUTION PROCESS 102
3C.10.3 PROBLEM TYPES 106

3C.10.4 PROBLEM RESOLUTION ADMINISTRATION 108
3C.10.5 HOW RESOLVED PROBLEMS LINK TO DEVELOPMENT ACTIVITIES 109
3C.10.6 CONCLUSION 111
3C.11 SYSTEM LIFE CYCLE MODEL 113
3C.11.1 INTRODUCTION 113
3C.11.2 BACKGROUND 115
3C.11.3 CCSAS CSE LIFE CYCLE MODEL 123
3C.11.4 CONCLUSION 129
3C.12.0 SYSTEM DEVELOPMENT 130
3C.12.0.1 INTRODUCTION 130
3C.12.0.2 REQUIREMENTS 131
3C.12.0.3 DESIGN 134
3C.12.0.4 ARCHITECTURAL ATTRIBUTES 136
3C.12.0.5 TECHNICAL REQUIREMENTS 137
3C.12.0.6 KNOWLEDGE MANAGEMENT 139
3C.12.0.7 CONCLUSION 142
3C.12.1 SYSTEM REQUIREMENTS ANALYSIS 144
3C.12.1.1 INTRODUCTION 144
3C.12.1.2 PROVEN PRINCIPLES 145
3C.12.1.3 PROVEN WORK PRODUCTS 150
3C.12.1.4 PROVEN TOOLS 152
3C.12.1.5 PROVEN PROCESS 153
3C.12.1.6 CONCLUSION 161
3C.12.2 SYSTEM ARCHITECTURE DESIGN 162
3C.12.2.1 INTRODUCTION 162
3C.12.2.2 PROVEN PRINCIPLES 163
3C.12.2.3 PROVEN WORK PRODUCTS 174
3C.12.2.4 PROVEN PROCESS 177
3C.12.2.5 CONCLUSION 182
3C.12.3 SOFTWARE REQUIREMENTS ANALYSIS 183
3C.12.3.1 INTRODUCTION 183

3C.12.3.2 PROVEN PRINCIPLES 184
3C.12.3.3 PROVEN WORK PRODUCTS 186
3C.12.3.4 PROVEN PROCESS 189
3C.12.3.5 CONCLUSION 197
3C.12.4 SOFTWARE ARCHITECTURE DESIGN 198
3C.12.4.2 PROVEN PRINCIPLES 198
3C.12.4.3 PROVEN WORK PRODUCTS 211
3C.12.4.4 PROVEN PROCESS 216
3C.12.4.5 CONCLUSION 220
3C.12.5 SOFTWARE DETAILED DESIGN 221
3C.12.5.1 INTRODUCTION 221
3C.12.5.2 PROVEN PRINCIPLES 221
3C.12.5.3 PROVEN WORK PRODUCTS 224
3C.12.5.4 PROVEN PROCESS 228
3C.12.5.5 CONCLUSION 237
3C.12.6 DATABASE DESIGN AND DEVELOPMENT 239
3C.12.6.1 INTRODUCTION 239
3C.12.6.2 PROVEN PRINCIPLES 239
3C.12.6.3 PROVEN WORK PRODUCTS 251
3C.12.6.4 PROVEN PROCESS 255
3C.12.6.5 PROVEN STANDARDS AND REVIEWS 263
3C.12.6.6 PROVEN TOOLS 264
3C.12.6.7 CONCLUSION 267
3C.12.7 SOFTWARE CODING 269
3C.12.7.1 INTRODUCTION 269
3C.12.7.2 PROVEN PRINCIPLES 270
3C.12.7.3 PROVEN WORK PRODUCTS 278
3C.12.7.4 PROVEN PROCESS 281
3C.12.7.5 PROVEN TOOLS 287
3C.12.7.6 CONCLUSION 292
3C.12.8 SYSTEM INTEGRATION 294

3C.12.8.1 INTRODUCTION 294
3C.12.8.2 TOTAL SYSTEM INTEGRATION RESPONSIBILITIES 295
3C.12.8.3 APPROACH TO COORDINATING CSE SYSTEM AND SDU SCHEDULES 302
3C.12.8.4 MANAGING CCSAS CSE INTERFACES WITH EXTERNAL ENTITIES AND THE SDU 308
3C.12.8.5 MANAGING EXTERNAL ENTITIES IN COMMON WITH THE SDU 324
3C.12.8.6 INTEGRATING CSE SYSTEM HARDWARE AND SOFTWARE 325
3C.12.8.7 CONCLUSION 327
3C.12.9 TEST MANAGEMENT 329
3C.12.9.1 INTRODUCTION 329
3C.12.9.2 PROVEN PRINCIPLES 330
3C.12.9.3 THOROUGH TEST COVERAGE 336
3C.12.9.4 STRUCTURED TEST APPROACH 340
3C.12.9.5 THOROUGH AND ACCURATE TEST DATA 346
3C.12.9.6 INTERFACES – EXTERNAL AND SDU 347
3C.12.9.7 REGRESSION AND PERFORMANCE TESTING 348
3C.12.9.8 INDEPENDENCE IN SYSTEM TEST 351
3C.12.9.9 INPUT INTO BUILD PLANNING ACTIVITIES 351
3C.12.9.10 INPUT INTO PROBLEM RESOLUTION PROCESS 352
3C.12.9.11 SYSTEM QUALIFICATION TEST SUPPORT 352
3C.12.9.12 CONCLUSION 353
3C.13.0 SYSTEM IMPLEMENTATION 354
3C.13.0.1 INTRODUCTION 354
3C.13.0.2 IMPLEMENTATION APPROACH 361
3C.13.0.3 UNDERSTANDING THE IMPLEMENTATION APPROACH UNDER THE TWO PHASES OF THE CCSAS CSE PROJECT 363
3C.13.0.4 CONCLUSION 367
3C.13.1 DATA CONVERSION 368
3C.13.1.1 OVERVIEW 368
3C.13.1.2 APPROACH 369
3C.13.1.3 CONCLUSION 391
3C.13.2 TRANSITION MANAGEMENT 392

3C.13.2.1 OVERVIEW OF TRANSITION MANAGEMENT 392
3C.13.2.2 CHANGE MANAGEMENT 394
3C.13.2.3 IMPLEMENTATION MANAGEMENT 422
3C.13.2.4 TRANSITIONING USERS AND FUNCTIONS 444
3C.13.2.5 ON-SITE SUPPORT 467
3C.13.2.6 LOCAL INTERFACES 473
3C.13.2.7 CONCLUSION 477
3C.13.3 USER TRAINING 479
3C.13.3.1 INTRODUCTION 479
3C.13.3.2 APPROACH 479
3C.13.3.3 CONCLUSION 509
3C.13.4 HARDWARE AND SOFTWARE INSTALLATION 511
3C.13.4.1 OVERVIEW 511
3C.13.4.2 OUR APPROACH 513
3C.13.4.3 MOVES, ADDS, AND CHANGES 528
3C.13.4.4 EQUIPMENT AND SOFTWARE SPECIFICATIONS 528
3C.13.4.5 RISKS AND MITIGATION STRATEGIES 532
3C.14.0 PRODUCTION SUPPORT 533
3C.14.0.1 CUSTOMER SERVICE 536
3C.14.0.2 CONCLUSION 556
3C.14.1 OPERATIONS 557
3C.14.1.1 METHODS, PROCESSES AND PROCEDURES 558
3C.14.1.2 OPERATIONS PROBLEM RESOLUTION 564
3C.14.1.3 METHODS AND TOOLS 570
3C.14.1.4 APPROACH TO DEVELOPING AND MEETING OPERATIONAL SERVICE LEVEL OBJECTIVES 584
3C.14.1.5 APPROACH TO TRANSITIONING OPERATIONAL RESPONSIBILITIES TO THE STATE 585
3C.14.1.6 CONCLUSION 587
3C.14.2 MAINTENANCE 588
3C.14.2.1 OVERVIEW OF THE MAINTENANCE STRATEGY 588
3C.14.2.2 APPLICATION MAINTENANCE 590

3C.14.2.3 HARDWARE AND SOFTWARE REFRESH CYCLE 593
3C.14.2.4 APPROACH TO TRANSITIONING MAINTENANCE RESPONSIBILITIES TO THE STATE 594
3C.14.2.5 CONCLUSION 595
3.C.14.3 HELP DESK 596
3.C.14.3.1 OVERVIEW OF THE HELP DESK FUNCTIONS, PROCESSES, AND PROCEDURES 596
3.C.14.3.2 APPROACH TO DEVELOPING AND MEETING HELP DESK SERVICE LEVEL OBJECTIVES.....608
3.C.14.3.3 APPROACH TO TRAINING HELP DESK STAFF 609
3.C.14.3.4 TRANSITIONING HELP DESK RESPONSIBILITIES TO THE STATE.....609
3C.14.3.5 CONCLUSION 610
PART 2 611
STATEMENTS OF WORK & STATE RESPONSIBILITIES.....611
PART 3 650
CONTRACT DELIVERABLE LIST (CDL) 650

3C: Technical Management Approach (TMA DID)

The alliance Technical Management Approach promotes and supports the widespread realization of the CCSAS project's technical and business goals.

3C.1 Introduction

3C.1.1 Purpose

The purpose of this response is to detail how the alliance will fulfill the requirements for the Technical Management Approach (TMA) as outlined in Exhibit 5L, Deliverable Item Description, Technical Management Approach.

The alliance technical management approach provided in the following sections describes the system engineering, development, and implementation activities that will be performed in creating the CCSAS CSE application – the next generation of child support systems. We discuss the processes, methods, tools, and experience the alliance will bring to bear in performing and managing system development activities throughout the project lifecycle.

3C.1.2 Document Overview

The overall description of the Technical Management Approach is found in section 3C.3 Overview of Technical Management Approach. The sections following 3C.3 address the specific requirements of the TMA DID, and how these are fulfilled or exceeded by the alliance approach to managing and executing technical activities.

TMA Contents

Attachment 3C is structured as required by the Technical Management Approach Deliverable Item Description. Our TMA consists of the following sections:

3C.1: Introduction

3C.2: Referenced Documents

3C.3: Overview of Technical Management Approach

3C.4: Configuration Management

3C.5: Requirements Management

3C.6: Technical Risk Management

3C.7: Knowledge Management

3C.8: Technical Reviews

3C.9: Technical Quality Management

3C.10: Problem Resolution
3C.11: System Life Cycle Model
3C.12.0: System Development
3C.12.1: System Requirements Analysis
3C.12.2: System Architectural Design
3C.12.3: Software Requirements Analysis
3C.12.4: Software Architecture Design
3C.12.5: Software Detailed Design
3C.12.6: Database Design and Development
3C.12.7: Software Coding
3C.12.8: System Integration
3C.12.9: Test Management
3C.13: System Implementation
3C.13.0 System Implementation
3C.13.1 Conversion Management
3C.13.2 Change Management
3C.13.2.1 Introduction
3C.13.2.2 Transitioning Users and Functions
3C.13.2.3 Implementation Approach
3C.13.2.4 Change Management
3C.13.2.5 On-Site Support
3C.13.2.6 Local Interfaces
3C.13.2.7 Conclusion
3C.13.3 Training
3C.13.4 Hardware and Software Installation
3C.14: Production Support
3C.14.1: Operations
3C.14.2: Maintenance
3C.14.3: Help Desk
3C.15: Appendices

3C.1.3 Definitions

See Rider G, Attachment G-1, Part 4 Acronyms and Glossary.

3C.2 Referenced Documents

Our technical management approach is based on proven practices and internationally recognized standards. This section provides a list of these external sources.

3C.2.1 Introduction

This section lists the number, title, revision, date, and source of external documents referenced within the TMA.

3C.2.2 Referenced Documents List

IEEE 828-1998, Standard for Software Configuration Management Plans, Jun-98, IEEE Standards Board

IEEE 1028-1997, Standard for Software Reviews, Dec-97, IEEE Standards Board

IEEE 1220-1998, Standard for Application and Management of the Systems Engineering Process, Dec-98, IEEE Standards Board

IEEE 1042- 1987, IEEE Guide to Software Configuration Management, IEEE Standards Board

IEEE 12207-1996, Standard for Information Technology - Software Life Cycle Processes, Jul-00, IEEE/EIA Standards Committee

IEEE 12207.2-1997, Standard for Information Technology - Software life cycle processes - Implementation considerations, Apr-98, IEEE/EIA Standards Committee

IEEE J-STD-16-1995, Trial-Use Standard for Information Technology Software Life Cycle Processes Software Development Acquirer-Supplier Agreement, Sep-95, IEEE/EIA Standards Committee

ISBN 0-201-70710-1, The Rational Unified Process - An Introduction, 2nd Edition, ©2000, Philippe Kruchten

3C.3 Overview of Technical Management Approach

Effective technical management can make or break a project. The foundation of the CCSAS project will be the well-engineered development, deployment, and maintenance of California's next generation child support system.

3C.3.1 Introduction

The alliance recognizes that the CCSAS CSE system provides indispensable tools for delivering Child Support Enforcement services to clients and as such is the significant long-term priority for DCSS. The proper planning and approach for conducting system development, implementation, and production support is critical to the success of the CCSAS CSE project. We realize that the best-laid plans can go awry without closely managing and monitoring system engineering activities throughout the system lifecycle.

In this section we summarize the alliance methods and approach for planning, managing, executing, and controlling major system engineering activities. The remaining sections within 3C Technical Management Approach provide more details for each portion of our approach.

3C.3.1.1 Technical Objectives

The alliance brings strong credentials in systems development and architecture object-oriented design of net-centric systems.

The alliance has a long history of successfully implementing large-scale highly complex systems that encompass the entire systems development lifecycle from project planning and startup through maintenance and operations. This experience, combined with the detailed architectural standards described in the SCP, has led us to the following high-level technical objectives and guiding principles:

- Offer a robust technical management approach based on a sound methodology and proven practices for planning, managing, executing, and controlling the project's engineering activities
- Conduct a Technical Management program in accordance with our Technical Management Approach
- Construct a Technical Management Approach that meets the intent of IEEE 12207
- Organize integrated and regular reviews between the alliance, FTB, and DCSS to help keep the project on track and on target, i.e., reduce surprises to either side

- Build an approach that will support design and architecture techniques necessary to create a next generation system
- Develop, implement, and monitor processes and procedures necessary to implement technical management plans
- Conduct a System Engineering program in accordance with our System Engineering Management Plan (SEMP) (CDL TM 001)
- Construct a System Engineering program that meets the intent of IEEE 1220
- Develop a configuration management program that meets the intent of IEEE 1042
- Perform maintenance activities and cycle processing to support the operation of CCSAS CSE environments
- Feature architecture principles and objectives contained in SCP Exhibit 3K

These technical objectives and guiding principles provide a foundation for building robust plans, executing the plans, and controlling the processes throughout the project so that we can deliver the next generation child support system for California.

3C.3.1.2 Certification – A Key Business Objective

Federal certification is a key objective for many reasons. The alliance understands the importance behind certification and has kept the following points in mind as we crafted our TMA to meet the needs of the CCSAS project:

- Support CCSAS Project Staff with their certification planning and review activities (SOW TM 0.6)
- Work with CCSAS Project Staff and federal reviewers to help plan and conduct financial test deck testing
- Provide technical assistance to assist CCSAS Project Staff in achieving ACF certification
- Assist CCSAS Project Staff with developing and implementing strategies to resolve certification issues
- Assemble the correct team to construct our TMA – a team with federal, state, and California child support experience, a team that understands the unique structure of California's operational environment, and a team experienced in large systems development.

3C.3.2 Proactive, Not Reactive

The alliance has proven abilities in creating systems that meet federal certification requirements. We have successfully led the effort in eleven states to achieve federal certification of child support systems. We bring this level of discipline to the CCSAS project and have incorporated our collective experience into our TMA.

We would like to begin with an overview of how we intend to structure development, introduce some of the thoughts behind our implementation and change management activities, and a description of how we will support and maintain the new system once it enters into production. This overview will also introduce our development methodology, our use of industry-wide standards and procedures, and our market-leading tools and

proven processes–all of which combine to create a firm foundation for a successful project.

3C.3.2.1 Primary Life Cycle Processes

3C.3.2.1.1 System Development

We propose Rational Unified Process (RUP) and component-based development for the project (see Figure 3C.3-1). RUP is a mature and proven framework and set of tools. It is focused on requirements-based modeling and is highly compatible with Java 2 Platform Enterprise Edition (J2EE) and WebSphere.

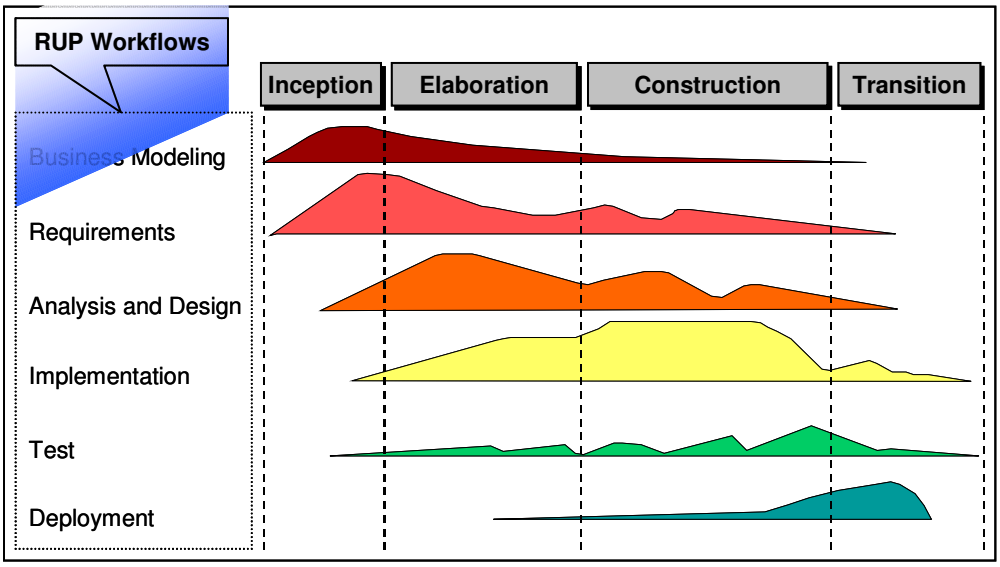


Figure 3C.3-1 Overview of Rational Unified Process – *RUP is a comprehensive, Web-enabled set of software engineering practices.*

But using component technologies requires much more than picking a programming language like Java or reading a book on UML and appointing oneself an object-oriented expert. These technologies have widespread impact on system development processes. They impact the approaches and techniques not only for coding, but also for design, architecture, methodology, work planning and project management, unit testing, team organization, and developer training. The alliance understands this, and we have been pioneering, defining, and perfecting component development processes for the last decade.

Some key development principles that we have incorporated into our approach are:

- Incremental development rather than typical waterfall delivery – capitalizes on lessons learned from earlier iterations
- Work cells – focuses skills on specific functional or technical capabilities and creates responsibility for end-to-end integration

- What-if thinking and team coordination – code reuse is designed in from the beginning, across teams
- Low-Fidelity prototyping – a proven way to address user buy-in and to confirm design concepts with the user community before investing in software development
- Requirements traceability and forward/reverse engineering – today's tools support greater levels of traceability than ever before, this further integrates the requirements definition, design, and coding process and enhances scope control

3C.3.2.1.2 System Implementation

The alliance has developed an implementation methodology that is ideally suited to the challenges of managing a complex system implementation in a diverse county-administered environment. The alliance CCSAS CSE implementation approach focuses on up-front planning to result in an efficient transition in the implementation sites. This approach has proven to provide a useful framework that increases the efficiency and quality assurance features of a standard methodology, yet also provides opportunities to customize the approach based on specific Local Child Support Agency (LCSA) characteristics. Figure 3C.3-2 shows that our methodology is comprised of seven implementation disciplines that work to ready a county for transition across four implementation steps.

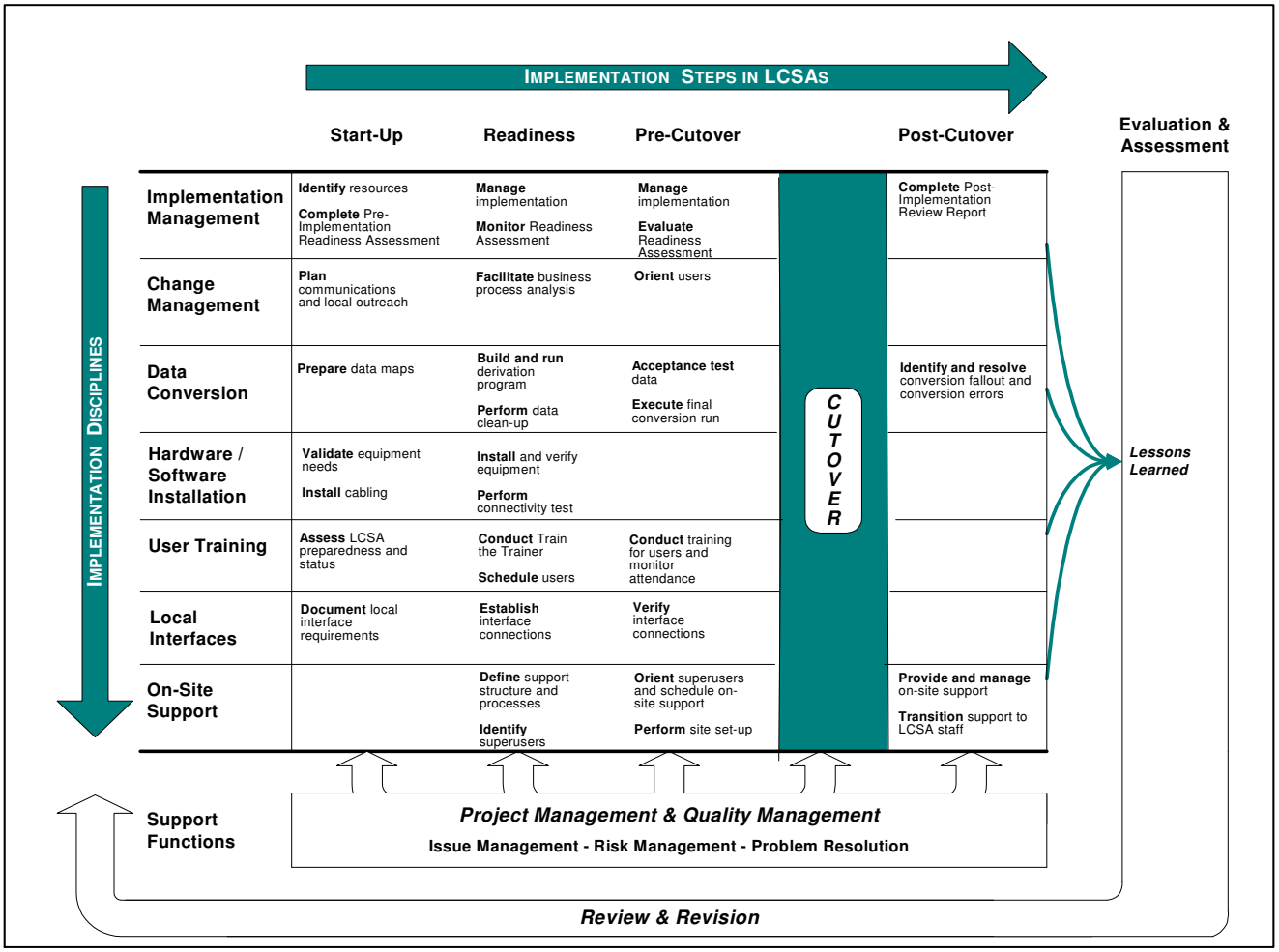


Figure 3C.3-2 Overview of Implementation Approach by Discipline and Steps - *The seven implementation disciplines work throughout the four implementation steps to prepare each county for transition.*

3C.3.2.1.2.1 Data Conversion

Data conversion presents a significant challenge in California because data must be converted from multiple legacy systems with different data structures and varying levels of data quality. As such, the alliance places emphasis on mitigating risk through the reuse of a conversion approach that already boasts a solid, long-standing track record within California. Our conversion approach is based on several guiding principles and has been tested and refined through the successful conversion of 35 counties to their current consortia systems:

- The complex data sets inherent in child support systems must be divided into discrete phases to simplify the end-to-end conversion process
- Data conversion must be done iteratively to map, edit, and thoroughly cleanse data prior to cutover
- Conversion edits and validations applied at cutover are standardized for consistency across converting counties
- Data mapping and derivation must be flexible to support differing LCSA legacy sources

- Successful data conversion requires coordination between DCSS, LCSA, and vendor staff

Our conversion methodology is supported by a toolkit that increases efficiencies throughout the conversion cycle. At the heart is the Automated Conversion System (ACS), a powerful tool for performing data edits and validations on data bound for the target system. In effect, the ACS serves as the "conversion bridge" into these systems by only allowing cleansed data to cross. To date, the alliance has successfully built and utilized an ACS for the ARS, CASES, KIDZ, and STAR/KIDS consortia.

3C.3.2.1.2.2 Change Management

Our national implementation experience, proven implementation methodology, lessons learned from the California CWS/CMS project, and extensive consortia system experience come together to implement the next generation child support system. Users and functions are transitioned in a way that adopts uniformity, moderates LCSA (and program) disruption, and mitigates risk. This section identifies and describes the steps necessary to carry out the alliance's two-phase implementation strategy:

Change Management – Change Management encompasses our strategy for adopting uniformity and managing the impact of change on users and external entities through business process analysis and a communications program which includes external outreach and user orientation activities. It describes the scope of change management and the issues considered in the development of our change management program – which is founded upon committed sponsorship; thorough understanding of business process changes; and strong communication and involvement principals.

Implementation Management – Implementation Management describes our implementation management approach, including how we organize the overall implementation team to execute our approach and how this approach works in the individual sites undergoing transition to the CCSAS CSE system. We also provide a definition of implementation roles and responsibilities, based on our experiences with similar implementation projects.

Transitioning Users and Functions – This section describes our strategy for transitioning users and functions, and the sequence in which the benefits associated with Version 2 and 2 may be achieved. It defines the timing of each transition along with the issues considered and the assumptions made in determining the timing of the transition. It further describes how our approach limits disruption to users and customers.

On-Site Support – We describe how our implementation methodology and consortia system experience minimizes user disruption by providing staff experiences in the new system during the critical period following cutover. This description includes the scope of support provided and the period of time support is available.

Local Interfaces – Local Interfaces describes our strategy regarding connecting to local county interfaces, including the welfare interface and county court interface. The section also describes the methods by which local interfaces are established and verified.

3C.3.2.1.2.3 User Training

The alliance training team is accomplished in both traditional and non-traditional methods of instruction, participating on many successful projects where classroom and

e-learning delivery methods were used to achieve the learning objectives of the user community.

As depicted in Figure 3C.3-3, our approach to training is aligned with the alliance's two phased approach to system implementation. During Phase I, the alliance will leverage the training program materials, method, delivery approach, and experienced resources already in place for CASES. For the CCSAS Version 2 application, the alliance develops training materials, methods, and a delivery approach tailored to the CCSAS CSE application and its users. The alliance will apply both traditional and non-traditional methods of instruction to deliver training during Phase II of the project.

Phase I - CASES Implementation		Phase II - Best of Breed CSE Application Implementation
CASES Statewide Services	***Child Support Application***	**New Best of Breed CSE Application**
14	***Number of Counties to be Trained***	**58**
LCSA Staff State Staff	***Trainee Groups***	**LCSA Staff State Staff**
Reuse of existing CASES Training Program materials, methods, and delivery approach	***Training Strategy***	**Custom development of all components required to support rollout of the new statewide CCCSAS system**
Delivery by LCSA Trainers	***Strategy Characteristics***	**CCSAS application used with e-learning course modules to deliver scenario based training Instructor-Led Classroom Training Coach-Assisted E-Learning at the Desktop Learning Management System Training Database**
Sunset CASES Training Program with rollout of Best of Breed CCSAS Application	***Post-Implementation***	**Maintain and Operate CCSAS Training Program Transition to State Control at End of Two Years**

Figure 3C.3-3 Overview of alliance Training Strategy – *User training provided by the alliance is focused on the Version 2 application.*

3C.3.2.1.2.4 Hardware and Software Installation

This section describes the strategies, methodologies, and tools required to procure and deploy approximately 10,000 desktops and peripheral equipment to approximately 84 LCSA and non-LCSA office locations throughout the State of California. A well-timed and well-planned installation of hardware and software is a key component of the overall alliance implementation strategy.

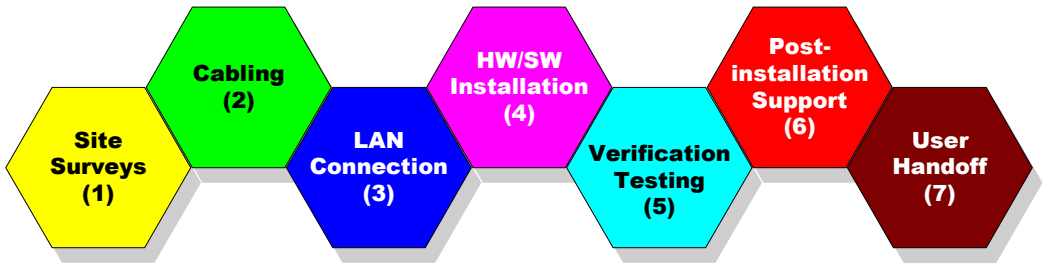


Figure 3C.3-4 HW/SW Installation Process. – *The process of hardware and software rollout begins with site surveys, followed by installation, verification, and, eventually, handoff.*

3C.3.2.1.3 Production Support

The alliance knows that operations support involves more than managing a customer's business applications to provide availability and responsiveness. As such, we have proposed a tightly integrated production support organization that will allow us to effectively support the application, environment, technical infrastructure, application users, and the customers of the Child Support Program. The key components of our overall production support approach include:

- Leveraging the existing infrastructure and proven operational procedures of an established data center
- Including experienced operations, support, and application maintenance personnel on the production support team so problems can be addressed quickly and end user impacts reduced
- Establishing and assisting with the staffing of a customer service center to respond to citizen inquires
- Providing IVR and eChild Support customer self service capabilities to program participants and stakeholders
- Establishing and maintaining control over CCSAS CSE environments by using processes and procedures have been defined and refined during thousands of projects
- Communicating system status and availability to application users
- Answering, resolving, and managing application user requests through a centralized help desk and enterprise-wide incident management system

Because the effectiveness of a production support organization is limited by the maintainability of the application, technical infrastructure, and environments they are given to support, our strategy is to design, build, and test maintainability into the CCSAS CSE environment from the start. Maintainability has been, and will continue to be, a major consideration of the alliance when making application design choices, and hardware and software selections.

3C.3.2.2 Supporting Life Cycle Processes

A life cycle model based solely on system development, implementation, and production support is incomplete. The foundation of the alliance life cycle model includes supporting processes as well. These supporting processes provide activities to help manage, develop, deliver, and support business capabilities. Support processes help incorporate quality into the system development process and the other primary life cycle processes discussed in 3C.3.2.1 of this section.

3C.3.2.2.1 Configuration Management

The objective of the configuration management process is to identify and manage configuration items through the development and deployment cycles of the system. Configuration items are soft items such as software classes, data elements, and documentation as well as hard items such as network interface cards, PC monitors, and routers. The CCSAS CSE configuration management process will be used to:

- Monitor and manage the development, build, and release of CCSAS CSE software components; and
- Monitor and manage the configuration, delivery, and installation of hardware equipment to central and remote operational sites for CCSAS CSE operations.

3C.3.2.2.2 Requirements Management

This section describes the process by which requirements are managed throughout development of the system. Requirements are the basis for which work is structured in a software development environment. Our approach for documenting and validating requirements is based on industry-leading tools, including DOORS for requirements management and Rational Rose for component and data modeling.

3C.3.2.2.3 Technical Risk Management

Technical risks are analyzed on an on-going basis to deal with changing conditions and priorities on the CCSAS CSE project. Our approach to technical risk management is to:

- Focus attention on mitigating risks – preferably resolving potential problems before they occur;
- Review identified problems to see if our current set of identified risks needs modification; and
- Analyze problems that have entered our problem resolution process to see if previously identified risks are appropriately mitigated.

3C.3.2.2.4 Knowledge Management

The alliance understands the importance of the development of key skills and the transfer of knowledge to State staff. At the completion of the CCSAS CSE project, the State will operate, support and maintain the application. Therefore, a knowledge management approach is vital to provide training to facilitate the development of knowledge and skills of the State staff. This section describes the knowledge management process for State staff that will work on the CCSAS CSE system.

3C.3.2.2.5 Technical Reviews

This section underscores the importance of formal and informal technical reviews. The alliance recognizes the importance of these reviews and why they are included in international standards like IEEE. Technical reviews are an integral part of many important development and implementation tasks of the CCSAS CSE project.

3C.3.2.2.6 Technical Quality Management

Technical quality management should encompass the entire development life cycle and include more than just a simple inspection process or a set of testing procedures. The alliance uses the ExPECT model throughout the life cycle to create specific processes and procedures that can be measured to gauge and enforce quality. The ExPECT model is a five-stage, customer-centered process that measures progress against clearly defined expectations. The model helps identify processes, measurements, and tracking techniques that will be used to enforce strong quality standards.

3C.3.2.2.7 Problem Resolution

Problems can and do occur and the alliance plans to have a robust problem resolution process in place to capture, evaluate, resolve, and control problems as they are discovered on the CCSAS CSE project.

3C.3.3 Conclusion

In this section, we summarized the alliance methods and approaches for planning, managing, executing, and controlling major system engineering activities. We developed our approach and methods for system engineering activities based on our extensive experience implementing large-scale government systems in Child Support as well as other program areas. Further, we tailored our approach to meet your specific needs as stated in the SCP.

The alliance approach, the cornerstone on which technical activities are completed, has been summarized here and is described in greater detail in subsequent sections. Our proposed technical management approach and methods position the project for success and contribute to the deployment of a system that meets the objectives of FTB and DCSS. The System Engineering Management Plan (TM 001) and the Software Development Plan (TM 002) – the two major technical deliverables – provide the direction in completing technical project activities using the approaches and methods prescribed by the alliance.

3C.4 Configuration Management

Configuration management is more than just source code control; it protects the integrity of project configuration items as they evolve through the development life cycle, from project launch through maintenance and operations.

3C.4.1 Introduction

The purpose of this section is to describe the management approach for configuration items created during development, implementation, and maintenance and operations of the CCSAS CSE system. Configuration management is part of an overall quality control program that will be implemented for this project to promote quality development and deployment of the application and computing environment.

The alliance understands that California's child support program must be able to adapt to change. Change could result from something simple to something major (e.g., federal changes like PRWORA). Therefore any approved changes in scope and requirements must be properly reflected in our system configuration and communicated to interested parties.

To this end, the alliance presents our proposed configuration management plan based in part on the following principles:

- Manage the system configuration in accordance with the Configuration Management Plan (CDL TM 003)
- Use an integrated data management system to capture and control configuration items
- Use a version control tool that employs check-in, check-out mechanisms.
- The State will have access to our version control tool.

The approach described in this section provide control of deliverables and work products throughout the project lifecycle – a critically important task for a project of the size and complexity of the CCSAS CSE project.

3C.4.2 Overview and Objectives

The objective of the configuration management process is to identify and manage configuration items through the development and deployment cycles of the project – maintaining the integrity of the configuration items throughout. Configuration items include soft items such as software classes, data elements, and documentation as well as hard items such as network interface cards, PC monitors, and wiring punchboards. The CCSAS CSE configuration management process is used to:

- Monitor and manage the development, build, and release of software components;

- Monitor and manage the configuration, delivery, and installation of hardware equipment to central and remote operational sites for operations
- Monitor and manage documentation

The alliance will establish and maintain a central repository to track configuration items. This repository contains comprehensive information about an item's type, usage, current status, and history. We will use a commercial configuration management product to provide high level and detailed information on configuration items. These tools provide automated management, access control, and review capability to CCSAS project members, both State and alliance, so that quick information on an item is obtained.

A Configuration Management Plan (CDL TM 003) will be created which defines the activities and responsibilities for creating and managing configuration items and overall CCSAS CSE configuration integrity. This plan will address software build planning and migration procedures and how they are affected by software testing and problem resolution. In addition, specific configuration management roles and processes will be created to maintain predictable, repeatable steps to move items through the development and deployment process.

3C.4.2.1 Roles and Responsibilities

The CCSAS CSE configuration management process requires a formal configuration management organization to identify and track items, determine change impacts, review changes, apply changes, and oversee the process. The specific roles for the configuration management organization described in Figure 3C.4.1.

Role	Responsibility
Configuration Management Coordinator	The Configuration Management Coordinator is responsible for day-to-day operation and planning of the configuration management process. This involves creating and managing correct, functioning formal configuration management procedures, directing management of the Configuration Management repository, and managing the change process with the Configuration Control Board. Specific responsibilities: • Managing change process with Configuration Control Board • Reporting on Configuration Management items • Facilitates compliance audits • Manage release packaging • Managing configuration item migration • Validating synchronization where appropriate between multiple site configurations
Configuration Management Reviewer	The Configuration Management Reviewer is assigned to review configuration changes.
Migration Coordinator	The Migration Coordinator works with the operations, implementation and development teams to plan creation and management of software releases, site creation, and site change management. This person is involved with the problem resolution process so that software resolutions are assigned to specific software builds. The Migration Coordinator also works with the problem resolution team to address emergency software changes so they can be expedited when appropriate. The conditions that must exist to qualify a change as “emergency” will be defined in the Configuration Management Plan.
Configuration Repository Custodian	Manages the configuration repository. Makes or directs changes. Conducts informal audits to verify that items are being correctly managed. Works with operations personnel to verify that backup copies of the repository are taken and stored property. Responsibilities: • Baseline configuration items • Package releases • Act on migration forms • Maintain Configuration Management service levels • Product Reports • Manage access control • Manage version control

Role	Responsibility
Configuration Control Board	Formal body that reviews and decides on configuration changes. During initial design and development, the Configuration Control Board is comprised of CCSAS CSE business and architectural leads, who will be primarily interacting with the development and implementation teams. Once initial deployment begins the Configuration Control Board takes on the formal role of providing configuration management for ongoing releases and sites.
Quality Management	A member of the Quality Management Team conducts formal audits on the Configuration Management process to verify proper functioning.

Figure 3C.4-1 Roles and Responsibilities – *Specific responsibilities are defined for each member of the configuration management team.*

The Configuration Management Plan (CDL TM 003) identifies detailed responsibilities for specific configuration management activities. Figure 3C.4-2 shows sample process steps and responsibilities for software configuration management.

Activity	Description	Environment	Responsibility
Overall			
Manage Configuration Management Process	Manage process associated with the Configuration Management process.	N/A	Configuration Management Coordinator
Manage Configuration Management Tool	Manage tools associated with the Configuration Management process.	N/A	Configuration Repository Custodian
Process			
CCSAS CSE Problem	Enter issue into Configuration Management tool	N/A	Test Team
Review, Update, and Assign	Review queued problems Determine technical solution Determine development schedule Assign appropriately Update request within Configuration Management tool.	N/A	Project Management
Requirements and Design Specifications	Gather business technical requirements. Develop the requirements and the solution to fulfill the requirements (if not already included in Problem Description).	Development	Requirements/Design Manager

Activity	Description	Environment	Responsibility
Technical Review and Walkthrough	Review the proposed solution with the Technical Team to determine if the solution fits the technical architecture.	N/A	Requirements/Design Manager and Technical Manager
Management Review	Review the solution and determine development schedule (for new functionality).	N/A	Chief of Development
Build, Unit, and Unit Integration Test	Build the solution Component test the solution Assembly test the solution	Development	Application Development Manager
Migration to Test and Quality Assurance	Migrate the code associated with approved changes to test environment in accordance with software build strategy.	Test	Migration Coordinator / Technical Manager
System Test	Review the completed solution Verify that requirements have been met.	Test	Chief of Application Testing
Migration to Production Simulation	Migrate the code associated with approved changes to production simulation environment in accordance with software build strategy.	Production Simulation	Migration Coordinator / Technical Manager
System Verification Test	Review the completed solution Verify that requirements have been met	Test	Chief of Application Testing
System Qualification Test	State Review of the completed solution State Verification that requirements have been met	Test	Chief of Application Testing
Management Review	Review the solution Obtain final sign-off	Test	Chief of Development
Migration to Production	Migrate code associated with approved changes to production environment in accordance with software build strategy.	Production	Migration Coordinator / Service Delivery Manager

Activity	Description	Environment	Responsibility
Confirm Successful Migration	Review changes in production environment.	Production	Application Development Manager

Figure 3C.4-2 Configuration Management Activities – *Specific steps are completed during software configuration management.*

3C.4.2.2 Configuration Identification

Configuration Identification is the process of selecting, identifying, and naming Configuration Items (CI). A CI is an aggregate of hardware, software, and/or documentation, developed and managed as a single item. A CI contains information that is created as part of the system life cycle. For the CCSAS CSE project, CIs will take the form of:

- Hardware, software, and data design specifications
- Implementation plans and configurations
- Software and data definitions
- Software and data elements
- Hardware components
- Software and data releases
- Software and data values and rules
- System Development Deliverables and work products (e.g., System Engineering Management Plan and Logical Data Model, respectively)

Each group lead (e.g., Requirements/Design Manager, Technical Manager) assigns responsibilities to configuration item identification and management. Only sufficiently mature items are submitted to CM. For a software item to be considered sufficiently mature to warrant CM control, the item must be tested to the point where stability has been validated. Documentation must also demonstrate a level of maturity in that it must have been internally reviewed. Once an item is identified as significantly mature, the responsible party submits a request to enter the item into configuration control. This is accomplished using the automated CM tool. The Configuration Management Coordinator and Configuration Management Repository Custodian act upon the requests. The Coordinator accepts submissions and provides approval. The Repository Custodian baselines the Configuration Items, which is described in Section 3C.4.4.1.

3C.4.3 Configuration Management Environments

As enhancements are made, problems addressed, and testing discrepancies resolved, configuration items take on different characteristics and reside in different locations. Because items may vary between environments as they are modified and tested, the alliance approach identifies and accounts for multiple environments.

3C.4.3.1 Identification of Environments

The following environments will be managed using the configuration management process:

- Development (e.g., Project LAN, design tool repositories, code repositories)
- Test
- Training
- Conversion
- Production Simulation
- Production

The alliance anticipates that there will be multiple versions of some of these environments based on the phased release implementation. The configuration management tools and processes are consistent from environment to environment. However, the access and authority for implementing changes to the baseline will vary from environment to environment, based on the amount of control dictated for each environment. For example, in the development environment, a programmer may be allowed to introduce changes to CIs with default approval. However, changes to the baseline in the production environment would require a formal request and approval.

3C.4.3.2 Identification, Control, and Status Accounting

3C.4.3.2.1 Identification

In designating CIs, the Configuration Management Coordinator will establish a numbering scheme to correlate the CIs and associated documentation. Configuration identification is performed by the appropriate functional area manager in coordination with the Configuration Management Coordinator and is maintained throughout the project. Configuration identification provides the basis for applying management control of the system configuration. It allows for isolated items to be controlled, their status to be tracked, and their configuration to be reported.

The purpose of selecting CIs is to manage their development and subsequently to manage their change. Each CI is a delivered item to which specific functional, performance, and physical characteristics are allocated. An effective configuration management program requires defining CIs and placing them under configuration control at the appropriate time – when a work product is less prone to change and thus requiring a change to the baseline. To control and manage CIs, each must be separately named and then organized. The identification task must achieve the following objectives:

- Unambiguous Name: A name that identifies the object unambiguously. Section 3C.4.6 describes our process for naming CIs.
- Comprehensive Description: A description that is a list of data items that identifies the Configuration Item type (e.g., document, program, data) that is represented by the object. The descriptor provides a generic description of the function or purpose of the object and its role in a future release. Descriptors must be carefully selected and approved by the Configuration Management Coordinator.

- Resource Identification: Resources are “entities” that are provided, processed, referenced, or otherwise required by the object. Data types, specific functions, and variable names are considered object resources.

The Configuration Items selected form the project baselines.

3C.4.3.2.2 Control

Configuration control is the process of controlling change. It includes the systematic evaluation, coordination, and approval or denial of proposed changes to an established baseline. Software configuration control begins with the establishment of the first baseline for the project. Configuration control continues through each succeeding baseline and for development projects, culminating with the establishment of the product baseline. For maintenance and operation, configuration control continues until the system is retired. This process is described in more detail in Section 3C.4.4.2.

Throughout the development life cycle, multiple versions of configuration items will be created. It is necessary for configuration management to address versioning, which can result from:

- Multiple Projects modifying code
- Shared Services supported across all projects
- Different testing and production environments

The alliance archive retains information about each revision and the original work product, allowing members of a development team to retrieve, modify, and return any revision of a work product in a safe, organized and consistent manner. As stated earlier, the alliance will use an automated tool which supports the version control needs of the project by providing check-in/check-out version control of source code, binaries, drawings, documents, and other items from project archives.

3C.4.3.2.3 Status Accounting

Configuration Status Accounting is the recording, monitoring, and reporting of changes to established baselines. Status accounting provides information about each change to those with a need to know. The purpose of status accounting is to answer the following questions:

- What happened?
- Who did it?
- When did it happen?
- What else will be affected?

The flow of information for configuration status accounting is tied to the identified tasks or CIs. Each time a configuration audit is conducted, the results are reported as part of the status accounting task. Each time a CI is changed, it must be tracked to validate that updates to the documentation are disseminated. The implementation of approved changes is tracked, and if problems occur, the appropriate CI (Requirements, Design, and Code) can be identified. Status accounting keeps management, software engineers and other project personnel apprised of project status.

3C.4.3.3 Configuration Audits

Configuration audits validate the completeness of configuration items and the maintenance of consistency across each environment. Audit processes serve dual purposes. First, they validate whether the configuration management baseline accurately reflects the physical implementation of system products. Second, they verify that the configuration control procedures of the configuration management system are being followed. Should breakdowns in the process be identified, it affords the opportunity to identify process improvements.

Configuration audits will be performed periodically to verify the configuration status accounting information. The configuration audit reduces the likelihood that unapproved changes have been inserted and demonstrates that the configuration management verification is valid. Issues discovered in configuration audits are documented and tracked to closure as defined in Section 3A.11 – Issue and Action Management Plan.

Configuration audits may be informal or formal. Informal audits are scheduled and conducted by the Configuration Repository Custodian. Formal audits are scheduled and conducted by the configuration management team. The Configuration Management Coordinator is notified and facilitates the review for the Quality Management Team. Configuration audit discrepancies discovered are prioritized and significant issues raised to the Configuration Control Board for resolution. Configuration audit reports are baselined and history is kept. The audit history is periodically analyzed to determine if repeated configuration problems exist.

Configuration audits occur throughout the life cycle to validate that configuration management processes are used and that related tasks are completed according to the Configuration Management Plan. Audits are conducted at least once per month and will vary based on project phase. For example, audits of the requirements baseline will occur more frequently at the beginning of the project than during the post-production phase. Likewise, audits of training materials are more likely to occur a few months prior to training delivery than after all users have been trained. The findings of each audit are documented via a memo and are presented to the Configuration Management Coordinator and Configuration Control Board. These reports will be stored on the project LAN and available to project personnel.

3C.4.4 Configuration Baseline Management

3C.4.4.1 Baseline Management

Baselines are used as a point of reference for tracking the status of CIs. They are established at a specific point in time. The baseline identification process involves identifying the CIs that compose each baseline and approving and releasing the items for use by those who require them. The baseline identification process is executed throughout the project life cycle to create the original baseline and any subsequent baselines that may be required. Establishing baselines allow project team members to have an accurate representation of the CIs in a given environment. Further, the project team can then track changes against the baseline for a given environment.

After the Configuration Repository Custodian establishes the baseline, project staff review the baseline to validate its accuracy. The Configuration Control Board ultimately approves the baseline. After the baseline is approved, any changes to the baseline

must be completed using the formal Configuration Change Control process, which is described in the next section.

3C.4.4.2 Change Control Process

The purpose of change control is to monitor and manage the addition, modification, or deletion of a configuration items. A configuration item is a work product which has been baselined. The alliance change control process is comprise of three primary activities:

- Logging configuration change requests
- Analyzing configuration change requests
- Tracking changes

The following diagram (Figure 3.C.4-3) depicts the process by which a configuration change request is used to modify a configuration item deliverable or requirement. Each change to a configuration item must flow through the established configuration control process.

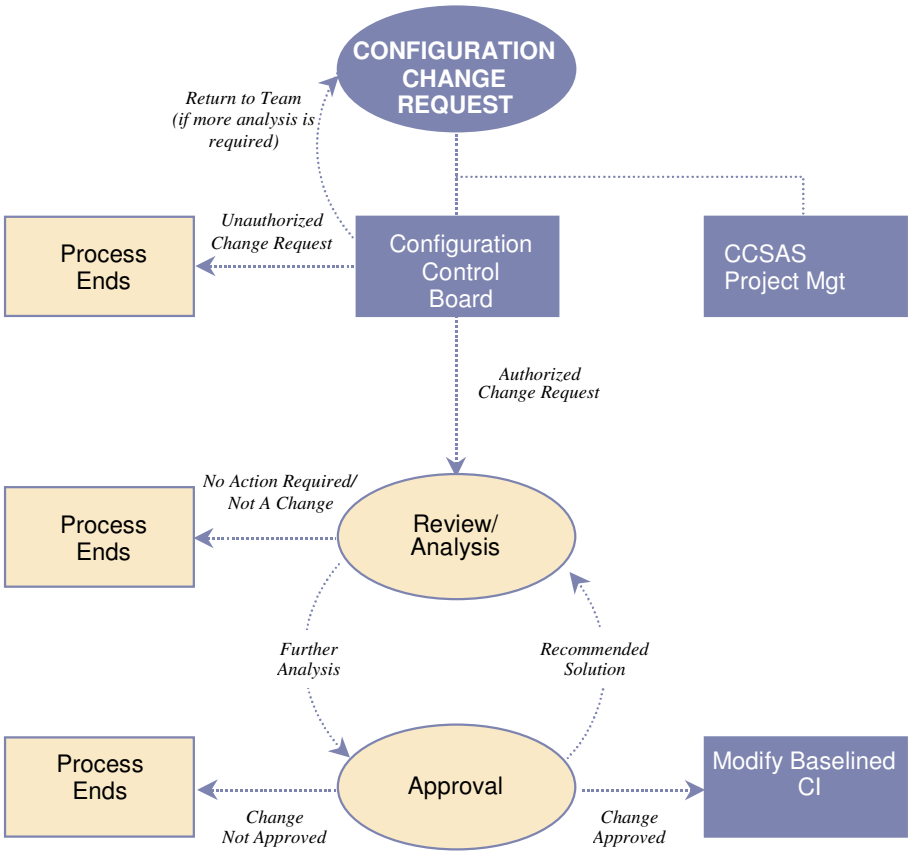


Figure 3C.4-3 Configuration Change Request Process – *Our configuration change request process maintains control over changes to the configuration baseline.*

A configuration item change request is triggered by work requests: additions, modifications, or deletions in functional capabilities, or problems detected during testing, verification, or validation (problems which correct errors but do not affect

scope/functional capabilities). Rather than using a paper-based form for recording and communicating change requests to the configuration management organization, the alliance will use Defect Manager of Mercury Interactive's Test Director product. The alliance will leverage the capabilities of this product to not only request changes to software configuration items, but technical documentation, and other non-software configuration items as well. Additionally, requests to enhance application functional capabilities will be captured with this tool.

The appropriate group manager and Configuration Control Board review each configuration change request and either accept, reject or approve it, or return it to the appropriate team for more detailed analysis. Based on the analysis, the Configuration Control Board management team decides whether to approve the configuration change request. Configuration Control Board disagreements on configuration changes are submitted to appropriate senior management for resolution.

Approved change requests are then prioritized by the configuration management team. When prioritizing changes, the configuration management organization works with group managers to assess the criticality of the change. For example, a problem that is adversely impacting caseworkers in the field would most likely be higher priority than a change associated with an enhancement that is not critical to workers delivering services in the field today. By factoring business need into the prioritization scheme, the alliance avoids impeding the release of critical defect fixes or other high priority changes. While business need impacts prioritization of configuration item change requests, it is not the only factor considered. Prioritization also considers

- Deadlines set forth in federal or state statutes that dictate when a change must be in place
- Opportunities for batching a change with other changes for efficiency purposes (e.g., multiple changes to the same application program)
- Other work items dependant on completion of such a change

In this manner, the configuration management team maintains control of software builds.

For approved configuration change requests, the configuration item modification is accepted and baselined by the Configuration Management Repository Custodian. If the configuration change results in a scope change, then a change request is forwarded to the Change Request Management process, which is described in detail in Section 3A.4 Change Request Management. Should the scope change request submitted to the Change Request Management process be approved, configuration item change requests for the appropriate items are created.

In preparation for project status meetings, a Configuration Change Request Report will be printed for distribution at the meeting. These regular status meetings provide a forum to inform project management of configuration change requests that could potential have scope implications.

3C.4.5 Database Items

3C.4.5.1 Database Changes

Database changes can have widespread impact on the application and those responsible for coding components, testing functionality, and training users on the

application. Because even the simplest database change can impact several configuration items and require coordination between multiple project resources, the alliance will control database changes through the same configuration management process used for hardware and software. In doing so:

- Database change requests are formally logged
- Requests are analyzed and the impact to other configuration items is determined
- Database changes are prioritized and coordinated
- Configuration item changes are tracked
- Changes are communicated

Database change requests will be formally requested and documented in the same manner software configuration change requests are recorded. The alliance will use Defect Manager of Mercury Interactive's Test Director to capture and send database change requests to the configuration management organization for analysis and review.

The impact of database change requests is assessed by the configuration management organization and the requesting organizational entity to identify the impact the change will have on other configuration items. In addition to application source code, other configuration items that may be impacted and analyzed include technical documentation, database backups of test data, and the various databases used within a given environment.

As described earlier, the group manager and Configuration Control Board review each change request and either accept it, reject or approve it, or return it to the appropriate team for more detailed analysis. Database change requests approved by the Configuration Control Board and management team are then prioritized relative to other configuration change requests.

Changes to configuration items including database design documentation, diagrams, and DDLs will be tracked using the configuration management automated tool. Changed configuration items are posted to the appropriate project archive by the configuration management organization and accessible to the project team.

3C.4.6 Technical Documentation

Technical documentation is controlled through the same configuration management processes as hardware and software. The configuration management tool will track documents used in the requirements analysis and design process. This allows the configuration management team to perform the following functions:

- Versioning and archiving technical documents – The configuration management tool will uniquely identify technical documents, such as requirements and design specifications. The tool will contain pointers to documents residing on the project LAN or other repositories, which is determined based on the CI identifier prefix;
- Change control procedures – The change control procedures for technical documents are the same as those employed for other CI types
- Logging configuration change requests using Defect Manager of Mercury Interactive's Test Director product

- Analyzing configuration change requests – Analysis of configuration change requests to technical documents are conducted in the same manner as for other CI types
- Tracking changes using an automated tool
- Notification procedures when technical documents are changed – Status accounting enables changes to technical documents to be posted and accessible to the project team

Naming conventions will be used to help identify the technical documentation generated during the course of the CCSAS CSE project. By doing so, unique identifiers are assigned to each configuration item. An example is: CCSAS-SDP-0024, which translates to a System Development Plan, uniquely identified with the number 0024. Figure 3C.4-4 identifies abbreviations that will be used for the technical documentation.

Document Name	**Abbreviation**	**Physical Location**
Configuration Management Plan	CMP	Project LAN
Conceptual Design Document	CD	Design Repository
Detailed Design Document	DD	Design Repository
System Development Plan	SDP	Project LAN
Test Plan	TP	Test Tool Repository
Test Procedures	PP	Project LAN
Test Results	PR	Test Tool Repository
Test Report	TR	Test Tool Repository
Product Specification	PS	Design Repository
Quality Assurance Plan	QAP	Project LAN
Requirements Specification	RS	Design Repository
Top-Level Design Document	TD	Design Repository
End-user Training Material	ETM	Learning Management Repository
Internal Team Training Material	TTM	Project LAN
Knowledge Management Plan	KMP	Project LAN
Risk Management Plan	RMP	Project LAN

Document Name	Abbreviation	Physical Location
Issue Management Plan	IMP	Project LAN
Glossary of Terms and Mnemonics	GTM	Project LAN

Figure 3C.4-4 Technical Document Naming Standards – *The nomenclature we employ for technical documentation is patterned after IEEE standards and identifies the physical location of the document.*

Naming standards will be developed, finalized, and communicated to appropriate personnel during the project.

3C.4.7 Conclusion

The size of the CCSAS CSE development effort and the complexity of the system being produced, from both architectural and functional perspectives, make the configuration management process critical to project success. Our proposed configuration management approach provides that configuration items are identified, organized and consistently controlled complete and correct, and are visible, traceable and verifiable. Our approach promotes quality and maintainability.

The alliance configuration management approach is both a process and a product quality control tool. The configuration management organization serves as the central control mechanism for changes to configuration baselines so the integrity and functionality of the application is maintained. Configuration management directs change and controls access to repositories to avoid conflicting actions that may jeopardize the integrity of the system and introduce problems.

We support system maintainability by establishing a way to identify changes to configuration items including application code, system hardware, and documentation. As such, the current status of any given configuration item is readily available. The ability to determine what has changed is one of the most powerful problem identification and testing aids. For example, when an application program is working one day and not the next, the first step is to find out what has been changed. By providing this ability, the alliance configuration management approach facilitates the timely disposition of system investigations.

Our approach identifies configuration items, maintain baselines, control changes to the baseline, perform status accounting, and periodically audit the process and products for compliance. Together, this process controls the integrity of configuration items throughout the project lifecycle.

3C.5 Requirements Management

Tracing requirements throughout the development process and managing changes to requirements are essential to the success of the project. Effective requirements management requires well-defined processes, appropriate tools, and an experienced management team.

3C.5.1 Introduction

The alliance understands that requirements will change – the baseline will evolve because of new child support program requirements.

This section describes the process by which requirements are managed during development of the CCSAS CSE application. Requirements are one of the first items collected and documented. Requirements are the basis for which work is structured and are a key driver of project scope.

IEEE defines a requirement as "a condition or capability needed by a user to solve a problem or achieve an objective". The alliance intends to work with State staff to thoroughly define requirements early in a project's lifecycle. No matter how thorough our mutual efforts may be to define requirements, inevitably the need for updates and changes occur. Controlling, managing and containing evolution of requirements is one of the most challenging issues faced by large projects. Therefore our requirements management approach not only addresses the initial collection of requirements, but also the change process for evolving requirements over a multi-year period. Project scope management is discussed in Section 3A.4 Change Request Management.

The objectives of requirement management are:

- Requirements should be collected through a formal, well defined process
- Changes can and should be initiated by any member of the project team, including those representatives whose interfacing systems may cause a need for change
- Any changes to the requirements are done so in a controlled manner to maintain a working baseline for software engineering throughout the life cycle
- The affected plans, deliverables, work products, and activities that are modified, remain consistent with the updated, baselined requirements

Our approach for documenting and validating requirements is based on industry-leading tools, including Telelogic's DOORS for requirements management and Rational Rose for component and data modeling.

The tools we use for managing requirements are market leaders in the industry; however, a tool is only as good as the discipline and experience to use it effectively. Our established design and development approach is founded on the assumption that project

tasks and work products are derived directly from documented requirements and can be traced back to them.

The following definitions will be helpful to provide clarity for the discussion in the remainder of this section:

Business Requirements are the requirements provided in the Business Requirements List. These requirements define the project's scope and are baselined at contract award.

System Requirements are the Business Requirements with additional detail added as necessary to provide clarification. As described in Section 3C.12.1, System Requirements are categorized as functional or technical, Version 1 or Version 2 (or both), and are documented in the System/Subsystem Specification (TM 016).

Software Requirements are the requirements stated at the lowest level of detail – that is, the requirements of software behavior. As described in Section 3C.12.3, Software Requirements are documented in the Software Requirements Specification (TM 020-1 and TM 020-2).

3C.5.2 Overview of Requirement Management Approach

Managing and tracking requirements is critical in complex projects. A successful, well-managed project is one where:

- Approved requirements are traced throughout the development process to verify that requirements are included in the final system
- No features are implemented unless they derive from approved requirements and receive the appropriate levels of review and control
- Suggested changes are evaluated for their effect on schedule, budget, resources, and risk
- Approved changes can be traced backwards to CCSAS Project Staff and user needs and can be traced forwards to design, coding, and testing

Features of our requirement management approach include the following:

Experienced Project Team Members – Capturing and tracking requirements is important to the overall success of the project. However, it is equally or more important to have knowledgeable, experienced staff on the project that understand the business of child support. Our project team members are experienced not only in large-scale system implementation projects, but more importantly, child support system implementations that result in Federal certification. They understand the complex State and Federal requirements and the detailed requirements that are a part of certified child support systems.

Proven Project Organization Structure – Our project organization places qualified personnel in management and decision-making positions. These personnel are experienced in both requirements management and project management methodologies. The organization stresses accountability, provides integration between teams, emphasizes planning to identify and avoid issues, and monitors status and work product quality.

Enterprise Capable Requirements Management Tools – The project team is supplemented with proven tools including DOORS and Rational Rose which trace requirements through the development lifecycle and measure the impact of requested changes on system components. Additionally, the requirements management tools are utilized to assist with consistency across project teams, including technical, functional, training, communication, and project management.

Formal Change Control Process – Changes made to requirements can have a major impact on system design and project scope. Our change control process has been proven on numerous projects similar to CCSAS in size and complexity and provides a formal process to measure the impact of requirements changes on the project prior to the change being implemented. Please refer to Section 3A.4 Change Request Management for more details on our change control process.

Acceptance Mechanisms – Formal change requests are considered accepted when the Change Control Board has approved the change request.

3C.5.2.1 Roles and Responsibilities

Various organizational entities are involved in requirements management, including staff from the alliance and CCSAS Project, as shown in Figure 3C.5-1.

Role	Responsibility
Change Control Board	• Evaluate/approve changes to the requirements baseline • Authorize adjustments to schedule or budget based on approved changes • Evaluate/approve baselining procedures
Configuration Management Coordinator	• Manage change process with Configuration Control Board • Report on Configuration Management items • Facilitate compliance audits • Manage release packaging • Manage configuration item migration • Manage synchronization where appropriate between multiple site configurations

Role	Responsibility
Requirements/Design Manager	• Develop and implement CCSAS CSE Requirements Management Plan • Administer the requirements change control process • Manage and coordinate the resolution of change requests • Monitor, trace and report on requirements and change requests • Elicit and document requirements • Facilitate requirements traceability • Transfer existing requirements into the requirements management software • Conduct interim reviews to discuss the status of requirements with project management, SCM, and quality assurance team members
Application Development Manager	• Participate in reviews with user representation to verify understanding of requirements • Analyze requirements to verify that they are feasible and meet the criteria for quality requirements • Analyze and evaluate change requests impacts on the schedule • Direct any activities required to incorporate accepted change requests into the software product • Identify technical issues
Chief of Application Testing	• Assess the impact of change requests on the testing planning and efforts • Analyze and evaluate change requests for testability • Incorporate accepted change requests into test plans • Identify issues that cannot be tested
System Implementation Manager	• Assess the impact of change requests on training and training materials
User Representative/Subject Matter Expert (SME)	• Define or assist in defining and documenting user needs, requirements, and user acceptance criteria • Review and agree to requirements before they are incorporated into the project • Provide a high degree of skill or knowledge pertaining to a specific business area or subject • Represent users and provides feedback to the user departments

Figure 3C.5-1 Roles and Responsibilities – *All entities are involved in requirements management at some point.*

3C.5.3 Managing Changes Prior to State Acceptance

The alliance and the CCSAS project staff will work together to capture the requirements for the new CCSAS CSE application starting with requirements as defined by the Business Requirements List at contract award. The alliance anticipates that additional

requirement change requests will be generated during the system requirements analysis process.

There are three points in time during the project when requirements are baselined, at successive levels of detail:

- Contract Award – Requirements defined by the final Business Requirements List provided by the CCSAS project staff and Appendix B of the Business Solution Response. This milestone baselines the Business Requirements.
- System/Subsystem Specification (SSS) – Requirements defined as part of system requirements analysis. Acceptance of the SSS baselines the System Requirements.
- Software Requirements Specification (SRS) – Requirements that are defined as part of the software requirements analysis. Acceptance of the SRS baselines the Software Requirements.

Requirement changes to the initial baseline are subject to the change request management process as described in Section 3A.4 Change Request Management. The SSS and SRS contain output from the scope change management process as well as the output from the requirements analysis phase of the project.

Telelogic's DOORS requirements management software has been selected as the alliance's requirements management tool. For each requirement, DOORS stores information such as requirement description, type, priority, source, and stewardship. A unique ID number is generated for each requirement, enabling consistent reporting as requirements are added, changed, and deleted.

3C.5.4 Documenting, Maintaining, Measuring, Reporting, and Controlling Changes

Since requirements form the basis for the project's work products and development activities, it becomes vitally important to maintain well-defined control over requirements.

The Requirements Management Plan (CDL TM 006) will be prepared to further define the details of the requirements management process to be implemented during the CCSAS project.

The alliance and CCSAS project staff will work together to incorporate existing change control procedures that CCSAS project staff utilize with those proposed by the alliance to reach a mutually agreeable set of change control procedures.

3C.5.4.1 Initiation of Requirement Change Requests

Initial requirements, once baselined, can only change when a requirement change request is initiated and subsequently approved.

Any member of the project team may initiate requirement change requests. This includes input from those representatives whose interfacing systems may cause a need for change.

Change requests are documented by the originators and submitted to the Requirements/Design Manager for initial review. Change requests are returned to the originators if they contain inadequate information for processing. Once a change

request is received it is evaluated for necessity, complexity and impact. The Change Control Board determines whether the project should accommodate the change request.

3C.5.4.1.1 Change History

The DOORS product supports the collection of change history and the use of requirement change requests to manage and track requirement changes. The history for a requirement change request will be updated based on decisions made by project executives and/or the Change Control Board. This history can be referred to as a reminder of the rationale behind earlier decisions and used to communicate to CCSAS Project Staff and users throughout the course of the project. Most change requests have proponents and opponents that may be interested in why the decision was made and who made the decision. This important history will help to reduce the need to revisit and rethink decisions.

Although not anticipated, history may be purged for product performance or readability reasons. If purging becomes necessary, the current requirements model could be archived. Then a new copy of the model could be created with the existing/previous requirement change requests and change history removed. Previous models, if any, would be retained and made available if needed. Historical information could be examined by opening previous model(s) as read only.

3C.5.4.1.2 Incremental Constraint

The alliance plans to incrementally constrain requirement change requests to prevent the uncontrolled initiation of requests and their associated costs to budget and schedule. The realization of project goals would be at risk if requirements were allowed to evolve without constraint over the course of the project. Requirements must be stabilized in order to develop a quality solution.

Constraint becomes progressively tighter as the project lifecycle unfolds. Requirement change requests are collected throughout the project lifecycle. Constraint is enforced at the first level of approval – the approval to proceed with impact analysis.

In general, changes introduced before state acceptance are considered for inclusion. After state acceptance, requirements will be baselined. Occasionally business needs force late changes in requirements. If CCSAS project staff need to initiate a change after the baseline has occurred, such changes are appropriately escalated as outlined in Section 3A.4 Change Request Management.

3C.5.4.2 Requirements Change Process

The Requirements/Design Manager screens requirement change requests and makes an initial determination regarding priority and status. The status code is used to monitor change requests as they move through the approval process. Priorities will determine the speed at which requests move through the approval process.

The Requirements/Design Manager is also responsible for coordinating the personnel needed to perform any necessary impact analysis. Impact analysis is deferred for any requests that are obviously out of scope (unless the request has critical impact). The Requirements/Design Manager may also coordinate with other project team members regarding feasibility and schedule impact and to further understand the effect on scope. Depending on the results of the analysis, the requirements management team

determines whether the request and appropriate information should be submitted to the Change Control Board for approval. As described in our Project Management Approach, the Board, composed of key members from the alliance and CCSAS Project Staff, meets regularly to review change requests. Requests that are approved are added to the requirements baseline.

3C.5.4.3 Requirements Management Tool

The alliance has selected Telelogic's DOORS, which has also been selected by the State, to maintain, measure, report, and control changes to requirements. DOORS is a leading requirements management tool, used by more than 50,000 users at over 1,000 companies around the world. It is designed to capture, link, trace, and manage a wide range of information. It has been used in the public and private sectors for managing issues, user requirements, system requirements, test conditions, strategic initiatives and business opportunities. This tool facilitates a project's compliance to specified requirements and standards.

The alliance plans to follow a rigorous requirements management process. As the Requirements/Design Manager receives new change requests, they are reviewed and entered into DOORS. If the change request is approved by the project's scope management process (as outlined in our Project Management Approach) the requirement becomes part of the baseline. If rejected, the status of the change request is updated as such and not acted upon. A history of change requests is maintained within DOORS.

DOORS also tracks metrics, indicated below, to determine the status of requirements management activities. The insights gained through an analysis of these measurements can be used to improve the existing process and to provide historical data for use in planning efforts. Additionally, these measurements can be analyzed to identify potential risks or trends affecting the project or to determine the project's progress.

The tool, for inclusion in status reports, can generate the following metrics:

- Number of change requests submitted
- Number of change requests approved
- Number of change requests by category (enhancement, addition, clarification, update)
- Number of change requests approved by category
- Number of change requests per functional area (functional, technical, common)
- Number of change requests approved per functional area
- Number of change requests per SDLC phase
- Number of change requests per requirement

Although DOORS and the other tools to be used provide many functions and features useful in creating and managing requirements, no tool by itself can deliver a well-managed project. The tool augments and supports the processes, standards, guidelines, and people that the alliance puts in place to gather and manage requirements. The alliance works with CCSAS project staff to determine the most appropriate way to use the right tools using the right process to achieve the requirements management objectives for the CCSAS project (CDL TM 006).

3C.5.4.4 Relationship Between Configuration Management and Requirement Management

The configuration management process identifies and manages configuration items through the development cycle – maintaining the integrity of the configuration items throughout. Early in the project, CCSAS CSE functional and technical requirements form the basis for the work to be completed throughout the remainder of the project. We recognize that during the System Requirements Analysis, necessary changes to the baselined requirements may be identified. These changes will be addressed through the Change Request Management process (as defined in Section 3A.4) and the configuration management process (as described in Section 3C.4). After the System/Subsystem Specification (SSS) is finalized (as described in Section 3C.12.1), the system requirements are established as the System Requirements baseline. Likewise, changes to the system requirements that may be identified later in the project must be approved through the formal configuration change request process.

Once requirements are baselined, changes can only occur using the configuration management process. Further, as new requirements are identified or as requirements change, the impact is likely to require software and/or hardware changes. The configuration management process governs changes to these configuration items as well. Requirements management is inextricably linked to the configuration management process.

As described in Section 3C.4 Configuration Management, should a configuration change result in a scope change, a change request is forwarded to the Change Request Management process (as described in Section 3A.4 Change Request Management). The interaction of these control processes (requirements management, configuration management, change request management) provides mechanisms to contain scope creep. It further contains the amount of change included in a single release. This fosters an environment of incremental and controlled change.

The procedural details for the interaction between requirements management, configuration management, and scope management will be addressed in the Requirements Management Plan (TM 006), the Configuration Management Plan (TM 003), and the Project Management Plan (PM 001).

3C.5.5 Assessing Impact of Requirements Changes

Ever-changing requirements can sabotage a project if not controlled and managed. Even when there is a process in place to document change requests, the project suffers if requests are approved without considering the corresponding impact on other requirements, project scope, project quality, schedule, budget, and resources.

Changes to requirements, especially in later phases of the development cycle, can be quite costly. Therefore, it becomes crucial to perform an impact analysis of each change request in order to assess the impact to related components, project scope, quality, and cost.

Changes to requirements usually result in schedule delays as well as increased time needed for redesign, software development, revisions of test plans, additional testing time and revisions of documentation. Furthermore, as a project progresses, the cost of simply assessing the impact of changes has impacts to the project schedule.

After change requests are submitted, logged, initially reviewed, and determined to be valid for the project, the detailed requirements are reviewed and analyzed by the team leads to assess the relative impact (impact analysis) of the change request on project scope, timeline, and budget. An overall impact severity level is assigned and a recommendation is formulated for the Change Control Board.

The Change Control Board has the responsibility for comparing the impact of a change to the CCSAS CSE system and its business value. Once the Change Control Board approves or rejects the change request, the results are logged into DOORS. Some change requests may be classified as enhancements and deferred for later consideration.

Software Tools

As discussed above, we plan to use DOORS to document, maintain, measure, report, and control changes to requirements. The DOORS requirements management software facilitates impact analysis. When a request is made to change a requirement, it is possible to identify related work products from design, coding, and testing.

For example, if a requirement changes, we can trace this requirement to a use case in Rational Rose. We can then trace from the use case to the corresponding use case realization. This use case realization traces to components, which reference classes in the UML class model, which map one-to-one to Java classes in the implementation model.

3C.5.6 Requirement Traceability into Life Cycle Products

While managing requirements documentation on a software development project is important, additional documentation created during the development of a project should also be managed and related, or linked, to requirements where applicable. The links between other project documents and the requirements afford the ability to know how changes affect specific project information that may be contained in various pieces of documentation. For example, a changed requirement affects not only design and program code, but test procedures and acceptance criteria as well.

Using DOORS allows our team to trace the relationships between requirements and other project work products. The first link to be established is between Business Requirements and the formal System/Subsystem Specification). Traceability extends from the SSS to the Software Requirements Specification (use cases). Links are also established between the use cases and subsequent work products from:

- Analysis
- Design
- Coding
- Testing

DOORS allows user-defined, multi-level traceability - for example, requirements to test and requirements to design. DOORS' Traceability Wizard generates link reports across many levels and displays them in the same view - providing lifecycle verification and validation.

It is theoretically possible to trace each line of code to the requirement that generated it. However, such detailed traceability requires a level of effort that far surpasses the benefit. The Requirements/Design Manager and Configuration Management Coordinator, in conjunction with CCSAS Project Staff, are responsible for determining the appropriate level of detail for the project, and for developing and enforcing the corresponding project standards and procedures.

We will utilize the Rational Unified Process (RUP) to analyze requirements, as shown in Figure 3C.5-2. High level use cases represent major business processes. Once use cases and actors are defined, sequence diagrams define specific processes within each use case. Using this combination of use case modeling and sequence diagrams, the system's functionality is documented.

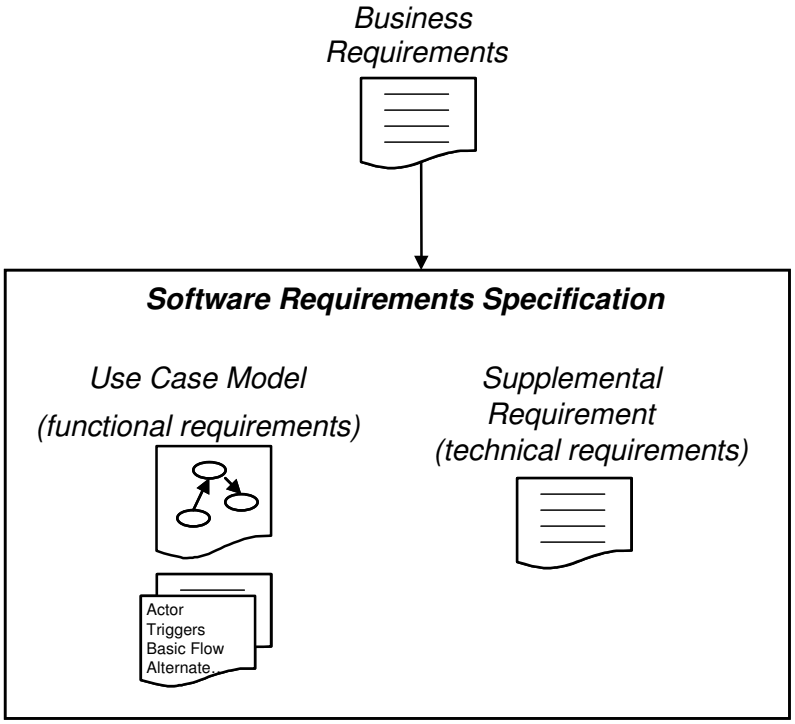


Figure 3C.5-2 Software Requirements Specification Process - *The requirements analysis process generates both functional and technical/quality requirements.*

Within the Rational Unified Process, use cases form the basis of subsequent design activities. Some work products are affected by more than one use case. For example, multiple use cases are supported by the same set of objects. Each use case adds or confirms the need for a particular behavior or attribute for an object. Figure 3C.5-3 illustrates the relationship between use cases and subsequent work products.

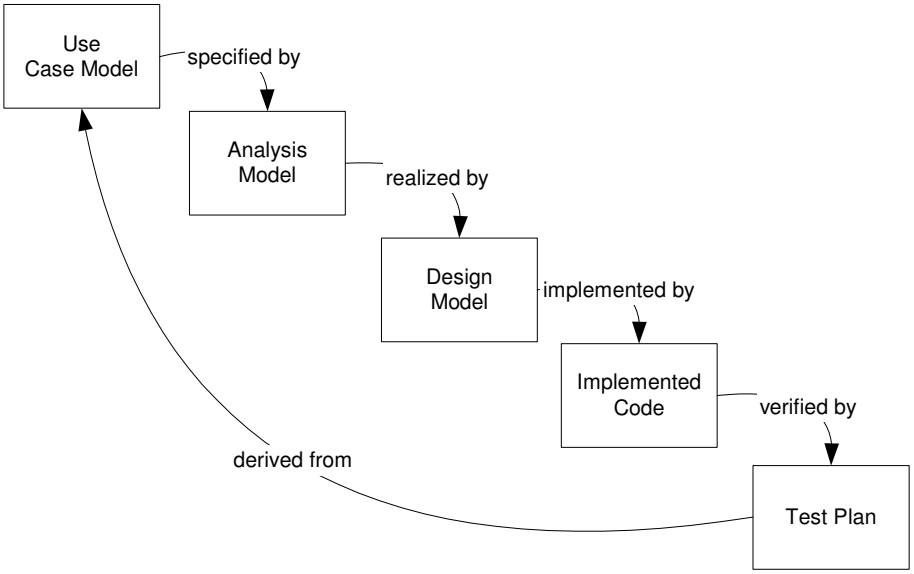


Figure 3C.5-3 Use Case Traceability - *Subsequent work products are derived from use cases.*

We supplement the standard RUP techniques with our own experience in object-oriented design, to establish naming standards, templates, and guidelines for work products. These tools, augmented with the proven software, make it possible to identify relationships between the requirements and subsequent work products.

Detailed traceability criteria will be addressed in the Requirements Management Plan (CDL TM 006) contract deliverable. More details on how the alliance plans to utilize RUP are contained in Section 3C.12.1 System Requirements Analysis and Section 3C.12.3 Software Requirements Analysis.

3C.5.7 Conclusion

Requirements are the foundation on which the CCSAS project is built. Managing requirements is one of the most challenging and important aspects of a large, complex project.

- The initial set of requirements must be clear and accurate
- A formal process must be in place to evaluate and approve any proposed changes to those requirements
- Tools and techniques must be in place to analyze the impact of a proposed change and manage the various versions of work products affected by a change

Standard, proven methodologies, well-defined processes, and proven tools are important. Our disciplined approach to managing requirements controls unwarranted changes which can be damaging to system development projects.

It is not enough to be able to read a book about design methodology, draw a process flow, or select a good tool. Requirements are bits of knowledge. Knowledge comes from people…experienced people. The CCSAS project staff know this and they have put together a strong, knowledgeable team. So has the alliance. We understand both the functional and technical requirements of this system. The alliance has brought together a team that knows next generation technology, knows child support, knows the California child support environment, and knows how to build the next generation in child support systems.

Our team will partner with the CCSAS project staff and utilize proven techniques in requirements analysis and change control management. Asking the right questions…deploying the right processes and tools…these are the steps to success!

3C.6 Technical Risk Management

The alliance provides the system implementation expertise to manage the technical risks required to successfully implement the CCSAS CSE system in California.

3C.6.1 Introduction

Large organizations, including government agencies, rely more and more on their IT systems. Many government agency processes are so dependent on their IT system that a failure causes the business process to stop. As a consequence, it is imperative to keep the CCSAS CSE application operating. This is where technical risk management becomes important.

IEEE standards state that it is critical to identify problems and mitigate risks continuously. They define risk management as "a continuous process for systematically addressing risk throughout the life cycle of a product or service". In keeping with that standard, the alliance has identified the following areas of technical risk to demonstrate our knowledge and experience in this area:

- Maintainability
- Manageability
- Availability and reliability
- Security
- Performance and scalability

The alliance uses a proactive and systematic approach for risk mitigation.

The alliance has a proven track record of implementing large, next-generation technology systems and mitigating the technical risks associated with these systems. A robust process that utilizes proven, market leading tools is a foundation for success. However, our experience has told us that it is important to identify risks as early as possible and tailor the process and tools to mitigate those risks. This section describes our approach for technical risk management – one that is proactive, systematic, integrated with project risk management, and ties to our problem resolution process. (This section deals with technical risks associated with System Development lifecycle processes.)

3C.6.2 Technical Risk Management Approach

3C.6.2.1 Roles and Responsibilities

Our technical management approach is based on the concept of continuous technical risk management defined by the Software Engineering Institute (SEI). We will team with CCSAS project staff to form Integrated Functional-Level Technical Risk Management Teams (IFTRs) who identify and assess the technical risks for specific technical areas of

the project. Risk items that are more global in nature are assessed and reviewed by the Integrated Project-Level Technical Risk Management Team (IPTR). Each technical risk management team is co-chaired by CCSAS project and alliance technical and/or management personnel to provide vigilance against emerging risk areas. Specific roles and responsibilities are outlined below in Figure 3C.6-1.

Role	Responsibility
Technical Risk Management Coordinator	• Maintain a Technical Risk Management (TRM) Plan • Approve TRM mitigating strategies • Track efforts to reduce technical risk to acceptable levels • Monitor problems entered into the Problem Resolution Management process to determine whether new risks can be identified or that previously identified risks are properly mitigated • Brief CCSAS CSE decision makers on the status of technical risk issues • Interfaces with the CCSAS Risk Management Committee for technical risk items
Project Level-Technical Risk (IPTR) Management Team	• Comply with approved Project Risk Management Plan for structuring an efficient and useful project risk management approach • Review and recommend to the project manager changes in the overall risk management approach based on lessons learned • Review and updates the global risk assessments • Supports the Risk Management Coordinator and project manager in carrying out their responsibilities
Functional Level Technical Risk (IFTR) Management Team	• Conduct risk assessments and develop risk management options and mitigation plans • Review and update the functional risk assessments • Confirm that information in the risk management database is current • Confirm that system development team responsibilities incorporate appropriate technical risk management tasks • Coordinate IFTR risk management activities with the IPTR

Figure 3C.6-1 Roles and Responsibilities - *Tracking and managing technical risks will help keep the CCSAS CSE application at the desired level of performance.*

3C.6.2.2 Problem Resolution and Technical Risk Management

Technical risks are analyzed on an on-going basis to deal with the changing conditions and priorities on the project. Both project technical management and problem management staff, as well as CCSAS project staff, work together to continuously identify, analyze, and mitigate technical risks and resolve problems throughout the various phases of the project.

Risk and problem management cut across project functional areas. Our approach to integrate Technical Problem Resolution and our Technical Risk Management is to:

- Focus attention on mitigating risks – resolving potential problems before they occur
- Review identified problems to see if our current set of identified risks needs modification
- Analyze problems that have entered our problem resolution process to see that previously identified risks are appropriately mitigated

3C.6.2.3 Integration with Project Risk Management

Proper technical risk management requires that risks be identified throughout the project development, implementation, maintenance, and operation phases. Technical risk management is not a one-time only activity. During project launch, it is integrated with our project risk management for the CCSAS CSE project.

Our technical risk management methods focus on internal technical risks (those under the control or influence of the functional development teams). Non-technical risks, or those external to the project, e.g. governmental legislation, are not addressed under technical risk management. Risks, no matter how they are classified, are managed under a single risk management plan as described in Section 3A.9 Risk Management.

Risk management team members share a common tool – PMOffice. PMOffice contains a risk management database to help project managers identify, prioritize, and communicate project risks in a flexible and easy-to-use form. PMOffice provides standard, integrated database capabilities to manage risks, as well as specialized functions for prioritizing and retiring technical risks and project risks. Risks are categorized to differentiate technical risks from project risks.

PMOffice has the capability to produce predefined or ad hoc statistical and historical reports. A set of standard or user-defined summary and detailed reports can be generated to review analysis of technical and project risk information. These reports allow the team to perform an assessment of the effectiveness of the risk mitigation strategies. Using a single, common tool for both project and technical risks, leads to better maintenance of risk information, allows for combined reporting if desired, and helps prevent information loss by merging two separate products or databases. (Refer to Section 3A.9 Risk Management for more detailed discussion of our overall risk management approach, including discussion of PMOffice.)

3C.6.2.4 Overall Risk Mitigation Approaches

Beyond the mitigation approaches to address the specific risk areas described in the remainder of this section, we use two techniques during the technology architecture design and build. These techniques - prototyping and technology verification - reduce risk for the remainder of the system development activities.

3C.6.2.4.1 Prototyping

Our technical architecture team builds a reference application as a means to reduce technical risk. The reference application is a “slice” of the CCSAS CSE application created for the purpose of verifying specific pieces of the application and technical

architecture; for validating our development methodology, tools, and procedures; and for acclimating our application developers to the CCSAS CSE-specific development environment and APIs. This prototyping activity establishes that the development environment and application developers are prepared to begin the application build process as well as proving feasibility of an approach.

3C.6.2.4.2 Technology Verification

We conduct an architecture test at the conclusion of the technology architecture build process before full-scale use by application developers to reduce the risks of re-work and low productivity.

3C.6.3 Technical Risk Identification, Quantification, and Mitigation

3C.6.3.1 Maintainability

Maintainability is the ease of maintaining or changing the developed system and/or its components over time. The ability to adapt the CCSAS CSE application to support new business functions is vital to comply with new legislation. Some of the major benefits of maintainability are:

- Increased system lifetime -The easier it is to fix system components and the easier it is to replace obsolete system components, the longer the lifetime of the system. The system can keep pace with changes in the business environment.
- Reduced costs - Costs are reduced through reuse and the reduction of repair times. For example, software maintenance can consume between 60 and 80 percent of the total expenditure on a typical product through its lifecycle, and over 50 percent of programmers efforts are dedicated to it.
- Higher availability - A reduction in the time to implement maintenance provides more availability and up time for users.
- Better reliability - Fixing problems quickly as they are encountered improves the reliability of products.

3C.6.3.1.1 Maintainability Risks

Potential maintainability risks may include:

- New parties unable to understand the functional and technical specifications of the system
- A system that is incapable of or difficult to upgrade to take advantage of new software and hardware technologies
- A system that cannot be modified for use in applications other than those for which it was specifically designed
- A system that is difficult to change
- A system that is time-consuming and/or costly to bring up at a new site
- A system that performs inconsistently and lacks standardization
- A system that has redundancy of software modules and little reusability

- Documentation and modules that cannot be traced through the development process - for example retrieving a version of the design documentation does not correspond to a specific version of a software module
- A system which cannot work cooperatively between heterogeneous environments

3C.6.3.1.2 Maintainability Risk Mitigation

Possible mitigation approaches are described in Figure 3C.6-2.

Mitigation Strategy	Approach, Methods, and Tools
Provide Architecture, Data, and Application Separation	Separation isolates the impact of change by verifying that architecture, business logic, and data are not mixed. • Use of standard APIs • Execution and application architecture framework validation • On-going reviews during development • Use of middleware to insulate business functions from dependencies on other system components
Establish a Layered Architecture	Layering allows independence from the underlying vendor's hardware/software by grouping and encapsulating different types of functionalities. • Use of standard APIs • Execution and application architecture framework validation • Use of middleware
Establish a Modular Architecture	Modularity facilitates analysis and increases the reusability of the functions. • Use of standard APIs • Component-based development • Use of middleware • Use of architecture hooks

Mitigation Strategy	Approach, Methods, and Tools
Consider Portability	• Portability requirements definition • Use of standard APIs • Use of middleware • Use of EAI tools • Use of CASE tools • Use of multi-channel architectures • Use of market leading vendor products • Technical documentation
Provide Configuration Management Tools	• Change control procedures and tools • Use of CASE tools
Adhere to Standards during Development	• Development standards • Establish review policy for each stage of development • Use of market leading vendor products
Prepare, Train, and Educate Maintenance Personnel	The maintenance personnel should be prepared, trained, and educated in order to effectively maintain and modify the architecture.
Provide Reusable Testing Models	• Automated test tools • Reusable test scripts, test data, and expected results
Use Proven, Reusable Architecture	By leveraging well-tested quality components from a reusable architecture, a project team can reduce overall project risk, and for longer-term projects, a reusable architecture can be used to deter changes in the underlying platform.

Figure 3C.6-2 Maintainability Risk Mitigation – *We propose a proactive systematic approach to mitigating maintainability risk.*

3C.6.3.2 Manageability

System manageability, or operability, deals with the ability to configure, administer and perform day-to-day IT operations on a system and its applications. Some of the major benefits of focusing on application and system manageability are:

- Empowered users - Increased efficiency and increased user satisfaction by enabling users to fully master their application. Users are more efficient by resolving issues related to the application themselves. They are not dependent on somebody else and therefore have a better sense of ownership and satisfaction with the system. In some cases, by mastering their applications, users will be more active in implementing new business processes, thus increasing the organization's competitiveness.

- Global reach - Application manageability is increasingly becoming important with net-centric applications; application users are often no longer part of the enterprise (they can be anywhere on the globe), and in most cases they are even anonymous (web site visitors). Therefore applications need to be designed to be easy to use (there is no way of training these external and anonymous users) and easy to operate (users are not physically within the enterprise boundaries - thus superusers and IT support can only be virtual).
- Lower operations management costs - Increasing manageability reduces the burden of supporting users as they perform application related operations such as error recovery, configuration and administration.
- Reduced dependency on vendor or system support - If the system is operable by means of everyday tasks, the need for having a vendor team or a large system support team to cope with ordinary tasks is reduced and as a result so is the cost.
- Increased availability - The simpler the functionality for start-up, shutdown, restart and backup, the lower the amount of unscheduled downtime. As a result, availability is increased. Additionally, functionality can be introduced into some systems that will allow them to report back to an operator about their status and resource utilization. This information can be used to proactively detect upcoming problems with the system, which can be addressed proactively without requiring any downtime.
- Standard skills required - If system's operations dialogues are straightforward and easy to understand the dependency on the operators to have deep knowledge of the underlying technology is reduced. As a result the skill requirements of the operators is reduced.
- Protected business assets - Many organizations heavily rely on their IT system to perform their business operations. The data and information contained in their systems is of critical importance to their business and to their customers. The loss of this data would have a large impact on the organization's ability to pursue their business. Having high manageability, and in particular operational backup and recovery procedures (even in case of disaster), significantly reduces the risks of data loss.

3C.6.3.2.1 Manageability Risks

Potential manageability risks include:

- A system that is difficult in managing error recovery, configuration, administration and parameterization
- A system that doesn't provide audit information regarding activity (update logs or application error logs)
- A system that is hard to configure according to user's need and preferences
- A system where recovery after an exception is complex and/or encumbering
- A system that is dependent on vendor system support to handle ordinary tasks
- A system that experiences unavailability due to unscheduled down time
- A system requiring highly skilled operators
- A system with high operations management costs

3C.6.3.2.2 Manageability Risk Mitigation

Possible mitigation approaches are described in Figure 3C.6-3.

Mitigation Strategy	Approach, Methods, and Tools
Understand System's Operations Characteristics	• System KPI analysis • Event type analysis • Scheduling requirements analysis • Recovery requirements analysis • Operations standards and guidelines for developers • Involvement of operators
Equip System to Handle Operations Requirements	• Identification of technical operations services • Enablement of service level measurements • Facilitation of initial installation • Involvement of operators
Prepare System to Handle Failure Situations	• Operations workflow analysis • Operations task automation • Backup/Restore strategy • 3rd party contracts and defined levels of service • Involvement of operators
Provide the Capability for Users to Manage the Basic Environment	• Table and parameter driven design • Application output management • Personalization
Provide Users with Tools to Handle Application Exceptions	• Application logging • Restart/Rollback mechanisms • State management services
Provide Flexibility within the User Interface	• Personalization • Internationalization

Figure 3C.6-3 Manageability Risk Mitigation - *We propose a proactive systematic approach to mitigating manageability risk.*

3C.6.3.3 Availability and Reliability

Availability and reliability are closely related and are the goal of automated systems. Availability is the ability of a system to be operating and accessible to its users when it is supposed to be and for the purposes for which it was designed and deployed. Reliability

is the ability of a system to consistently behave in a predictable manner both under normal and abnormal operating conditions.

Increased availability and reliability have cost considerations and may result in added systems development cost and added technology infrastructure investment. These costs must be balanced against quantifiable added value and intangible benefits when creating a business case.

Some of the major benefits of focusing on availability and reliability for the system are:

- Service delivery tailored to users - The service can be delivered when the users can benefit from it.
- Global access – Availability and reliability can be stretched to support global access. For example, with a net-centric business capability users can reach a company from anywhere on the globe at times convenient to them.
- Increased user satisfaction and productivity - User goodwill can increase if the CCSAS CSE application offers consistent availability and reliability. Users will develop a strong perception that the system will meet their expectations consistently. When they develop a confidence in the CCSAS CSE system, their efficiency could increase, along with satisfaction and productivity.
- Continuation of business activities – A CCSAS CSE system providing continuous availability to users is critical in preventing loss of service for clients.
- Increased availability – A CCSAS CSE system that requires less downtime, thus increasing its availability to its users. CCSAS will use cluster technology for the server back-end. This type of technology is often used for providing high levels of reliability and consequently a high level of availability.
- Decreased maintenance – A CCSAS CSE system that requires less maintenance because of the reliability designed into it. For critical business activities, it is worthwhile making higher investments in building reliable systems from the beginning, since the maintenance efforts for these types of systems must be of very high priority and thereby more costly than normal maintenance.

3C.6.3.3.1 Availability and Reliability Risks

Potential availability and reliability risks include:

- A system that experiences downtime or lack of availability under stressful conditions
- A system that is inaccessible due to failure of one or more components
- A system that delivers incorrect results
- A system that delivers unexpected results
- A system that compromises, corrupts, loses or delays information
- A system that is not robust and fault tolerant of invalid input
- A system that experiences failures caused by faults of its components
- A system that provides different results when a given set of inputs is reprocessed

3C.6.3.3.2 Availability and Reliability Risk Mitigation

Possible mitigation approaches are described in Figure 3C.6-4:

Mitigation Strategy	Approach, Methods, and Tools
Understand System Availability Requirements	• Business performance modeling • Capacity modeling and planning • Disaster/recovery planning
Confirm Architecture and Application Components Support the Availability Requirements	• Data distribution • Segmentation and replication • Architecture component selection • Application distribution • Redundancy and failover • Load balancing • Error handling • Modularity • Batch process design
Secure the Technology Infrastructure	• Physical and logical security • Redundancy and failover • Disaster/recovery planning
Coordinate Operations Procedures to Enable Availability	• ISP agreements • Capacity modeling and planning • Systems management monitoring tools
Confirm Availability Requirements are achieved Through Testing	• Capacity modeling and planning • V-model concepts • Performance testing
Confirm Quality Support for End Users	• Service desk organization and procedures • On-line user assistance • Operations level standards for developers
Understand System Reliability Drivers	• User observation and involvement • Prototyping • Use of application and architecture standards
Enforce Standardization to Increase Reliability	• Use of application and architecture standards • Stage containment • V-model concepts

Mitigation Strategy	Approach, Methods, and Tools
Provide Consistency Under Normal and Abnormal Operating States	• Use of application and architecture standards • Prototyping • Usability testing • Data cleansing in conversion • Use of consistent error handling strategy
Confirm Technology Infrastructure Supports Reliability Requirements and Expectations	Services provided by the execution architecture, which support and interact directly with the application, should be developed using the same design techniques and standards as the application.
Test for Consistency and Integrity	• V-model concepts • Fault prediction analysis techniques • Performance testing • Negative testing, i.e., how the system behaves under abnormal situations

Figure 3C.6-4 Availability and Reliability Risk Mitigation - *We propose a proactive systematic approach to mitigating availability and reliability risk.*

3C.6.3.4 Security

There has been a need for security since the creation of computers. For example, access to the first mainframe computer was controlled by a password. Since that time, technology has advanced and so have the security threats to the systems using that technology. Net-centric solutions present a new set of security issues and risks. These include the following:

- The geographic distribution of critical resources, such as information and logic, has increased the risk of having resources accidentally or fraudulently altered at either their remote location, or while being transmitted between the remote locations and central site.
- The business capability's openness to newer users or interconnected systems has made the interfacing mechanisms more fragile by giving access to faceless users and/or to systems whose management is beyond the control of the central system management, making them by definition non-trustworthy.
- The increasing complexity of solutions with heterogeneous technologies and third-party products has in turn increased the probability that the final business capability has vulnerabilities.

All the technology and use evolutions have been accompanied by increased risks and the requirement for more sophisticated and powerful security solutions.

3C.6.3.4.1 Security Risks

Potential security risks include:

- Operation failure due to loss of passwords to malicious intruders

- Loss of child support monies due to false transactions caused by intrusion
- Noncompliance to legal regulations governing personal information such as, medical records, and social security numbers, etc.

3C.6.3.4.2 Security Risk Mitigation

Possible mitigation approaches are described in Figure 3C.6-5:

Mitigation Strategy	Approach, Methods, and Tools
Plan Security Up Front	• Follow security methodology • Involve security expertise • Perform security risk assessment • Leverage security framework • Develop security awareness • Evaluate security regularly
Apply Authentication and Identification	• Establish user ids and passwords • Establish public key infrastructure
Identify Appropriate Access Methods	• Utilize role-based access control • Deploy firewall for network access control
Protect Sensitive Information	• Utilize cryptographic services
Implement Accountability Mechanisms	Accountability provides the ability to correlate events with systems, processes, and users and to provide proof of transaction (non-repudiation). • Maintain audit trails and security event logs • Establish event monitors • Establish public key infrastructure
Establish Integrity Controls	• Establish virus protection program • Deploy intrusion detection system • Perform vulnerability assessment
Develop Controls to Enforce Level of Service	• Develop backup and restore process • Develop disaster recovery plan • Develop intrusion response plan • Deploy security patches

Figure 3C.6-5 Security Risk Mitigation - *We propose a proactive and systematic approach to mitigating security risk.*

3C.6.3.5 Performance and Scalability

Performance and scalability are closely related system attributes that can impact the business value of the system. Performance is the ability of a system to accomplish a given task in the timeframe needed to enable users to complete their business activities in a reasonable manner. Scalability is the ability to expand a system with reduced change in current viewpoints of the system or its procedures to accommodate growth (of transaction volume, data storage volume, concurrent users, etc.) Some of the major benefits of focusing on performance and scalability for the CCSAS CSE application are as follows:

- Increased user satisfaction and productivity - A high performing system directly increases end-user satisfaction, especially for on-line transactional systems, where the system response-time directly influences the efficiency of the user in performing business tasks.
- More adaptability to growth - A clear benefit of a scalable system is the ability to handle increases in data, network traffic and transaction volume with a reasonable investment as the CSE program grows (system components, implementation effort and lead time). Building scalability into the CCSAS CSE application allows for a quicker reaction to an increased number of users.
- More economical in the long term - Investing in design up-front has proven to be more economical in the long term than re-building an entire system when faced with business increases. It would be more expensive to re-build the CCSAS CSE system than to add, as required, the appropriate components that support business growth. For example, if new access channels are needed, net-centric can support multiple electronic access channels to disparate sources of information that are reachable by a large variety of users (computer, mobile, phone or notebook based).
- More services offered - New services can be offered to CCSAS CSE customers as new policies or programs are introduced. For example, high-speed networks based on ATM or SONET technologies support different types of traffic such as video, voice or data. The associated throughput capability represents tremendous opportunities for CSE to offer new methods of multi-media services to its customers. This can provide services for non-English speaking participants, new training media for caseworkers, and a variety of other ways to enhance current programs and services to both the local LCSA and its customers.

3C.6.3.5.1 Performance and Scalability Risks

Potential performance and scalability risks are:

- Technology infrastructure that becomes a cost burden over time due to poor performance
- Oversized hardware and software solution to manage caseload growth that generates high costs for unused capacity
- Poorly managed vendors that lead to low performance due to insufficient level of service provisioning and high repair times

- Third party dependencies outside the control of the system owner; for example, internet performance, is not accounted for in the design of the system
- Business disruption from performance degradation due to underestimation of the growth of the volume of business information to be handled by the CCSAS CSE system
- System performance that deteriorates and does not met performance requirements over time
- An IT system support organization unable to monitor and tune performance.

3C.6.3.5.2 Performance and Scalability Risk Mitigation

Possible mitigation approaches are described with Figure 3C.6-6:

Mitigation Strategy	Approach, Methods, and Tools
Identify and Understand Performance and Scalability Requirements	• Capacity modeling and planning • Performance modeling • Technology prototyping
Maintain Data-Related Performance and Scalability Focus	• A component-based n-tiered architecture allows for close to linear transactional scalability • Capacity modeling and planning • Database design • Data distribution and replication • Data physical storage optimization • Parallel processing • Performance testing
Maintain Network-Related Performance and Scalability Focus	• N-tiered architecture • Capacity modeling and planning • Event management • Network dimensioning • Performance testing
Maintain Transaction-Related Performance and Scalability Focus	• N-tiered architecture • Capacity modeling and planning • Parallel processing • Load balancing • Performance testing

Mitigation Strategy	Approach, Methods, and Tools
Plan for Peaks and Growth	• Performance modeling • Performance testing • Bottleneck handling
Maintain Performance Focus through System Life Cycle	• Application tuning • Database tuning • Network tuning • System software tuning

Figure 3C.6-6 Performance and Scalability Risk Mitigation - *We propose a systematic and proactive approach to mitigating performance and scalability risk.*

3C.6.4 Conclusion

Continuous technical risk management requires that risks be identified throughout the development, implementation, maintenance, and operation phases. We employ a proactive approach to technical risk management integrated with project risk management and tied to our problem resolution process.

3C.7 Knowledge Management

The alliance has a structured knowledge management approach. Alliance and CCSAS project staff work side-by-side to share knowledge and develop skills. State staff attend formal, informal and on the job training. The approach includes processes for measuring and enhancing the knowledge management activities.

3C.7.1 Introduction

The alliance understands the importance of the development of key skills and the transfer of those skills and knowledge to CCSAS project staff. At the completion of the CCSAS project, the State will operate, support and maintain the application. A knowledge management approach is vital to provide training to facilitate the development of the knowledge and skills of the CCSAS project staff. Knowledge management focuses on technical knowledge and skills, related to the specific needs of the individual roles and responsibilities. Training in non-technical fields, for example development of leadership skills, is not included in the scope of knowledge management. The alliance brings a proven methodology and the collective experience to facilitate knowledge management of the CSE application. This section describes the process for CCSAS project staff – the individuals who will help develop, operate, and maintain the CSE system. Training focused on end-user performance is addressed in Section 3C.13.3, User Training.

Our knowledge management approach provides a structured means for establishing and delivering training to CCSAS project staff in the use of the appropriate hardware and software development platform, tools, processes, and methodologies used during the project. Knowledge management activities align with the system development life cycle so that CCSAS project staff participates in learning activities relevant to their responsibilities. The approach involves the providing a recurring opportunity for the transfer of knowledge and information between the CCSAS project staff and the alliance team.

Our approach is based on working with the CCSAS project staff. State and alliance resources are co-located and work side-by-side throughout the project. This makes for a win-win situation for both the State and the alliance. Resources participate in project activities jointly. The alliance draws from State expertise to analyze design and develop the system. The CCSAS project staff perform project activities and receive training and coaching in order to prepare to support and operate the system when the project ends.

It is anticipated that CCSAS project staff require various levels of training, ranging from extensive to minor. The balance is achieved by matching the expectations of specific roles and responsibilities (as documented in a Knowledge Management Plan) with a personalized learning curriculum (expressed in an Individualized Development Plan). The alliance assumes the State is responsible for developing the baseline skill sets for the CCSAS project staff. Our approach provides ongoing training and learning activities

to develop the skills and knowledge specific to the CCSAS CSE technology and functionality.

With a proven track record in systems development, the alliance recognizes that child support systems are complicated to build. It is critical to include training throughout the system lifecycle to tool each member of the team with knowledge and skills to overcome this complexity. We also recognize that child support, because of its political and social importance, is a dynamic business where both state and federal policies will dictate ongoing changes to the CSE system. At the completion of the CCSAS project, the State will operate, support and maintain the application. Without a proper knowledge management program, CCSAS project staff might be left in a frustrating situation where they are unable to keep up with the requested system changes.

The alliance understands the importance of the developing key skills and knowledge in the CCSAS project staff. We propose a Knowledge Management approach that provides opportunities for relevant training for the individual.

Throughout the process of designing, building, and implementing the application, CCSAS project staff will be involved. The alliance proposes a project organization that is 'work cell' based. Each work cell is focused on a particular function or technical aspect of the system. Each cell shall include one or more CCSAS project staff member in its make up. This integration facilitates informal and on-the-job training as synergy is developed and fostered among team members. Our Knowledge Management approach also includes formal training in the tools and procedures that we propose.

3C.7.2 Challenges

A thorough Knowledge Management approach addressing the training needs of CCSAS project staff is important to the success of the CCSAS CSE implementation because of the following challenges.

- The Consortium Maintenance and Operations staff supports the current CSE systems, and it will be a change for DCSS to support a centralized statewide CSE system
- Child support applications are complex by nature. As such, it requires specific and deep knowledge in many functional areas, like case processing, and financial calculation and distribution
- The scope of the CSE application includes leading edge software and hardware. This will require a new set of skills and understanding of new methodologies. For example, rapid development and Rational Unified Process (RUP) are very different from traditional design, build and test approaches

The alliance estimated the knowledge management activities based on 47 people .. The approach is based on four key categories of project functions and within each category are project positions included in the proposed knowledge transfer.

- Systems Development and Operations/Systems Management and Planning
- Database Support
- Architecture Support

- Configuration Management
- System Administrator
- Technical Support
- Systems Development and Operations/Case Management Development and Operations
- Case Management System Design and Development
- System Analyst
- Case Management System Testing
- Case Management Conversion
- Systems Engineering
- Technical Liaison
- Requirements Engineering
- Process Engineering
- Transition Management

CCSAS project staff from within Systems Development and Operations and Systems Engineering will participate full time on the project team. We understand Systems Management and Planning personnel perform ongoing support activities while Case Management Development and Operations personnel perform functional design, development and testing activities. Systems Engineering personnel provide guidance during design, testing and implementation of information technology solutions. These resources require training to enable their participation in the development and on-going maintenance of the CSE system.

Individuals from within Transition Management have been moved to new roles and will develop new skills that help support the on-going operations and maintenance activities for the application. The transition management roles are not pertinent to the support activities that will be conducted after the system is transitioned to the State because implementation will be complete. To develop these new skills, the resources will work with the application development team members to confirm design and perform testing activities. They will also participate in the hands-on training and classes that have been recommended.

The alliance brings extensive experience in project management, system design, development, transition management, deployment, and support. Based on our experience and the specifics of the CCSAS CSE project, including the technology and functionality, we have identified key categories of skills and the related project positions within each category. CCSAS CSE project staff will be assigned to one or more project positions throughout the lifecycle of the project to facilitate skill transfer.

The alliance suggests the following skill categories and positions for CCSAS CSE project staff to participate in knowledge transfer:

Project Management

- CCSAS CSE Executive Project Director

Functional Architecture

- Data Architect
- NetCentric (Web and Component) Architect

Application

- Business Analyst
- Application Designer
- Data Access Developer
- User Interface Developer
- Java Developer
- Batch Developer
- Integration Architect
- Testing Lead

Database

- Database Designer
- Database Administrator (DBA)
- Database Support

System

- System Administrator

Infrastructure Architecture

- Network Architect
- Infrastructure Architect
- System Management Architect
- Security Architect

Hardware

- PC Support

Rider H includes the position names, the roles and responsibilities for each position, and the knowledge and skills needed by project phase.

The remainder of the Knowledge Management section describes our approach in greater detail. The section contains the following topics:

- Structured knowledge management approach
- Knowledge management curriculum
- Production support and project turn-over

3C.7.3 Structured Knowledge Management Approach

Our knowledge management approach provides a structured process for building skills through learning activities and providing recurring knowledge exchange. Transferring training knowledge, tools, and processes develops ownership of the solution and improves the solution's ongoing viability beyond the development phase.

The alliance will designate a system development team member as the Knowledge Management Lead. Our overall approach to knowledge management is to develop a plan that lays out the project organization, positions and skills and then develop detailed plans to provide the opportunity for the transition of knowledge. Specific learning activities to build skill sets based on defined role descriptions and required skills consist of a combination of formal training, informal training, and on the job training.

Figure 3C.7-1 illustrates our Knowledge Management Approach, which provides the step-by-step process for knowledge and skill development, and information sharing.

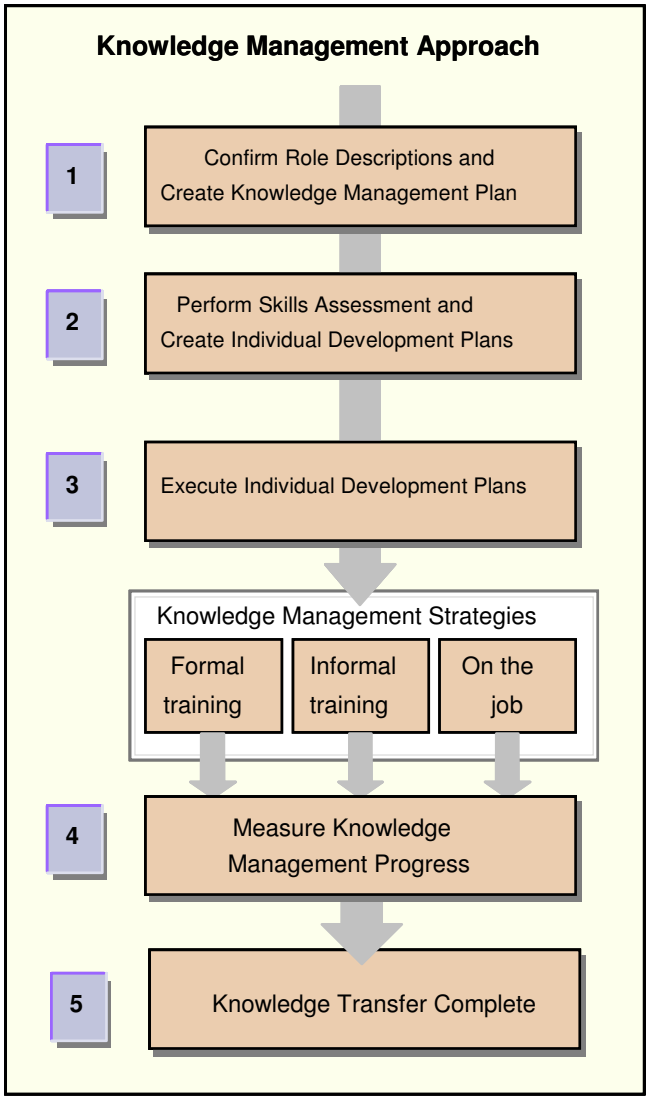


Figure 3C.7-1 Knowledge Management Approach - *Our knowledge management approach provides a structured process for building skills and sharing knowledge between CCSAS CSE project staff and alliance resources. Refer to Section 3C.7.3.5 for types of training.*

3C.7.3.1 Confirm Role Descriptions and Create Knowledge Management Plan

The first step of the Knowledge Management Approach consists of the confirmation of new roles and responsibilities and the development of the Knowledge Management Plan (CDL TM 007).

The objective of developing role descriptions and positions early at the start of the project is to identify the skills sets and coordinate them with appropriate positions. The Knowledge Management Lead works with DCSS and FTB leadership to determine and confirm these roles and responsibilities based on CCSAS CSE project staff skill sets.

The Knowledge Management Plan incorporates information from a consolidated list of the required knowledge and skills. The Knowledge Management Plan provides a learning plan for developing skills and building knowledge of the CCSAS CSE staff. The Knowledge Management Plan is used through out the life of the project as the road map for knowledge and knowledge transfer activities.

The Knowledge Management Plan will be delivered consistent with the timeframes in the CCSAS CSE Project Plan The Knowledge Management Plan includes:

- Identified staff roles
- List of required skills and knowledge
- List of knowledge transfer activities
- Time estimates per course
- Time estimates per resource

3C.7.3.2 Perform Skills Assessments and Create Individual Development Plans

In the next step of the knowledge management approach, the skills of State staff are assessed and Individual Development Plans (specific to the knowledge transfer activities) are created. The content of the Individual Development Plans is driven by the project roles and responsibilities. The precise balance of training will be achieved via matching the expectations of specific roles and responsibilities with a personalized learning curriculum expressed in an Individual Development Plan. The State is responsible for developing the baseline skill sets for the CCSAS CSE project staff.

Each team member starts the project with a different set of skills and skill levels. Working with their supervisors, CCSAS CSE project staff members self-assess the skills they bring to the project and identify any gaps that exist between those skills and the future skills required. This allows them to create an individual plan targeting the specific skills they need to develop.

The Individual Development Plan describes the specific training courses, learning activities and work tasks geared to the skills and capabilities of each staff person. Figure 3C.7-2 illustrates an Individual Development Plan template.

Individual Development Plan	
Section I	
Plan Begin Date **Participant:**	**Plan End Date** **Staff:**
Project Role:	
Knowledge Transfer Goals:	

Section II: Training Courses

Description	Due	Completed

Section III: Learning Activites

Description	Due	Completed

Figure 3C.7-2 Individual Development Plan Template - *The Individual Development Plan describes the specific training and work activities needed to develop the skills of each staff member.*

The Individual Development Plan documents for each CCSAS CSE staff member their role, knowledge management goals and knowledge management activities. Knowledge management goals are performance-based objectives with defined metrics in order to measure progress towards meeting the goals. For example, one knowledge management goal for a system developer may be to gain the ability to program using Java with minimal supervisory guidance by a specified date.

The CCSAS CSE project staff, with input from their supervisor and the Knowledge Management Lead, create their Individual Development Plan for a six-month timeframe.

3C.7.3.3 Execute Individuals Development Plans

After the creation of the Individual Development Plan, the next step is the execution of the learning activities identified in the Knowledge Management and Individual Development plans. This entails participation in:

- Formal Training
- Informal Training
- On-the-job Training

Figure 3C.7-3 illustrates samples of formal, informal, and on the job training that will be available during the CCSAS CSE project knowledge transfer activities.

TYPE	TRAINING MECHANISM	DESCRIPTION
Formal	Instructor-led classroom	Training presented by vendors, alliance and specialists on the project team in order to develop skills. Training is formal, structured, and presented in a classroom during predetermined and set classroom hours.
Formal	Self-paced computer-based training	Computer-based or web-based training delivery is in both synchronous and a-synchronous modes. Training is provided at the student's desktop. Training time is predetermined and scheduled.
Informal	Brown bag or Lunch and Learns	Learning sessions, scheduled over lunch, are in presentation and/or facilitated discussion formats. Interaction amongst attendees is encouraged. Sessions held on a regular basis.
Informal	Team Demonstrations	Ad hoc demonstrations by project teams of tools or functionality. Typically small group of attendees, not project wide. Sessions held on an ad hoc basis.
On the job	On the job training	Working with experienced resources, learn by doing, match work activities with learning objectives outlined in Individual Development Plans.

Figure 3C.7-3 Learning Channels - *Various channels for skill development and knowledge building are available to State staff during the project.*

Information about these delivery mechanisms is described in greater detail in Section 3C.7.3.5. After each formal learning activity, State staff evaluates the training. The evaluations provide feedback and enable the alliance to identify and incorporate changes to better serve the CCSAS CSE project staff.

3C.7.3.4 Measure Knowledge Management Progress

The alliance is committed to continued improvements to the knowledge management process during the CCSAS CSE implementation. It is important to track the progress of the Knowledge Management Plan and the achievement of Individual Development Plan goals.

The purpose of measuring knowledge management is to:

- Monitor knowledge management activities that are taking place throughout the life of the project and identify improvement opportunities
- Benchmark existing skills and knowledge in order to determine the most appropriate knowledge management activities for each person
- Provide a generic indicator of project progress in the area of knowledge management

Success of the knowledge management process is measured by reviewing the overall success rate of the Individual Development Plans as well as training evaluations. Our approach includes checkpoints to assess progress on an individual basis and project level.

Staff members and their supervisors revisit the Individual Development Plan every six months, and complete an Individual Development Plan Assessment survey. The survey provides the mechanism for State staff to assess the effectiveness of training, learning activities, project tasks, team interaction and project management support in meeting their knowledge management goals. The alliance uses this information to review knowledge management progress and recommend appropriate revisions and enhancements.

The Knowledge Management assessment survey form is shown in Figure 3C.7-4.

CCSAS CSE PROJECT
California Child Support Automation System

Individual Development Plan Assessment

Name of participant completing survey: _____

Dates covered by Survey: _____

PART A

Rate your experience with the CCSAS Projec t Knowledge Transfer process using the following criteria:

Exceeds (E): Impact of subject area on knowledge transfer gained has **greatly contributed** to meeting goals specified in my Individual Development Plan.

Meets (M): Impact of subject area on knowled ge transfer gained has **adequately contributed** to meeting goals specified in my Individual Development Plan.

Does Not Meet: (DNM): Impact of subject area on knowledge transfer gained has **not adequately contributed** to meeting goals specified in my Individua l Development Plan.

N/A: Category rating is **not applicable** for this survey period.

Mark an "X" in the appropriate boxes.

	DNM	MEETS	EXCEEDS	N/A
Training Courses				
Project Plan Tasks				
Alliance Participant(s) Interaction				
Project Management Support				
Overall Progress				

Please place additional comments on reverse side

Individual Development Plan Assessment

Figure 3C.7-4 Individual Development Plan Assessment Survey - *Learning assessment occurs throughout project to measure progress towards meeting defined goals.*

Once progress toward goals is assessed, the CCSAS CSE project staff update and create an Individual Development Plan for the next six months timeframe. New goals are created and new learning and work activities are defined.

In addition to tracking knowledge management on an individual level, the State will want information on the overall progress of knowledge management on the project. Cumulative information regarding the success of individuals and training evaluation

feedback is gathered and reported to State leadership. The State provides feedback on the efficacy of the knowledge management program.

3C.7.3.5 Types of Training

Knowledge management includes the different types of training - formal, informal, and on the job training. These different channels provide multiple means for learning the optimum approach for meeting individual learning styles. The types employ standard training techniques to accommodate various learning processes. The following paragraphs describe each type of training in more detail.

3C.7.3.5.1 Formal Training

Once the Knowledge Plan and the Individual Development Plans are completed, the CCSAS CSE project staff attend the formal training courses. These courses relate directly to the skill and knowledge requirements to develop, implement, and manage our proposed solution. Training addresses hardware and software development platform, tools, processes and methodologies.

Formal training consists of third party vendor training and alliance-developed training courses (including instructor-led and web-based training). Below is a list of the proposed formal training types:

- Third party vendor training – Vendor training in a classroom environment used to develop skills in the software and hardware being utilized. This training uses the third party vendors' certified trainers, courseware and training materials.
- Alliance developed training – CCSAS CSE project staff are provided with information (e.g., fact sheets, white papers, procedures documents) on project-specific methodologies, procedures, and standards. CCSAS CSE project staff receive training that bridges the gaps between vendor-provided training and specific knowledge requirements of the business solution.

The alliance provides the training materials for the software development tools, reflecting the customized usage and implementation of the software development platform, tools, processes and methodologies. The alliance follows a proven instructional systems design methodology to develop customized training (see Section 3C.13). The tasks include developing learning objectives, designing the course sequence of key content points, identifying the types of practice exercises, developing the course content, testing and producing the materials. These courses shall be repeatable, as defined in the project plan, throughout the life of the project. These course materials are maintained so they remain current and specific to the system development approach.

The following is a sample list of materials during Formal Training:

- Training manuals
- Operation instruction guides
- Job aids
- Participant guides
- Practice exercises

- Demo accounts
- Scripts
- Site maps

3C.7.3.5.2 Informal Training

Project environments are typically dynamic and eventful. Environments where constant learning and knowledge sharing are encouraged often characterize successful projects. The informal learning opportunities are the practical expression of encouraging and displaying learning so that proven practices can be quickly shared and implemented across the project.

CCSAS CSE project staff gain knowledge by attending informal knowledge-sharing events such as “Lunch and Learn” and team demonstrations. The alliance conducts informal working sessions with CCSAS CSE project staff to discuss various topics, such as the implementation plan, or see demonstrations.

Samples of training materials used during informal training are:

- FAQs (Frequently Asked Questions)
- Lessons learned documents
- Troubleshooting guides
- Glossary of terms

3C.7.3.5.3 On the Job Training

On the job training refers to the development of skills and knowledge through observation and hands-on experience. On the job training is a vital component of our knowledge management approach. On the job training occurs throughout the analysis, design, development, coding, integration and testing phases. To facilitate this type of training, the project teams are co-located throughout the project.

On the job training includes:

- Job assignments
- Shadowing
- Coaching
- Independent performance

Based on the position, skill level, and specific knowledge base, CCSAS CSE project staff are assigned to a specific functional or technical area with specific work activities. They initially learn by shadowing more experienced personnel who perform the work. The alliance assigns a peer mentor to each CCSAS CSE project staff member to help facilitate knowledge management. As skills develop, CCSAS project staff members begin to perform work activities with coaching from the experienced personnel. As they continue to increase their knowledge and build skills, the CCSAS CSE project staff members perform the work tasks more independently, and with minimal support.

The on the job training activities increase knowledge development and transfer as alliance and CCSAS CSE project staff work closely together to design, build and maintain the system. By participating in building the system, CCSAS CSE project staff gain system knowledge and development skills that are the foundation to maintaining and supporting the system.

3C.7.4 Production Support And Project Turn Over

This section address knowledge transfer activities during production support and project turnover. Knowledge management activities occur throughout the design, development and integration phases primarily through alliance provided formal, informal and on the job training. During production support prior to turnover, more extensive knowledge transfer activities including third-party vendor training occur to help prepare the CCSAS CSE project staff to operate and support the CSE application independently. This allows the State staff to develop skills on a stable system.

3C.7.4.1 Production Support

During production support, the alliance and CCSAS CSE project staff monitor system operations and fix issues encountered. The production support phase begins when the Version I Statewide Services functions are deployed. During this phase, CCSAS CSE project staff participate in system maintenance and support. CCSAS CSE project staff may need to sharpen their skills, gain knowledge of functional areas developed by other members, or learn new skills needed specifically for the maintenance of system components, before they can fully support the system without assistance from alliance members. This will occur through on the job training.

The alliance proposes to operate the application in a hosting center in San Jose. The production support phase for the CCSAS CSE system begins at initial deployment. During this period, the alliance provides a support team at the San Jose facility and continues to work with CCSAS CSE project staff in the project location. CCSAS CSE project staff may travel from Sacramento to San Jose or alliance staff may travel from San Jose to Sacramento to co-locate these teams.

On the job training occurs in the following ways:

- Informal instruction in the knowledge and skill area
- Shadowing the alliance members in support and maintenance tasks such as logging and tracking problems
- Performing the tasks with coaching and support from alliance members. The State team member may log issues, work on fixing problems, track progress on fixes, and update problem log with support
- Performing the tasks independently with reduced support from the alliance member

Formal training courses for production support personnel are provided during this time period. The knowledge transfer activities occur throughout the two years of production support - providing a comfortable timeframe for knowledge transfer.

During this period, we also recommend that CCSAS CSE project staff participate in cross-training activities through job rotation. That is, the maintenance responsibilities of

specific components rotate among CCSAS CSE project staff. This cross training supports the objective that the continued operation of a system is not dependent on any one individual. However, the rotation should not be so frequent that it disrupts the orderly maintenance of systems.

3C.7.4.2 Project Turn Over

Final knowledge transfer as a component of the overall knowledge management approach is designed to facilitate the smooth transition of knowledge, skills, documentation and operations from the alliance to CCSAS CSE project staff. Final knowledge transfer occurs as part of the overall system turnover process. Preparation for system turnover entails estimating the staffing requirements for CCSAS CSE project staff for ongoing maintenance as well as transferring responsibilities and related documentation from alliance to CCSAS CSE project staff.

Throughout post-production support, alliance and CCSAS CSE project staff work together to operate and maintain the system. Prior to turnover, the staffing estimates for State resources are reevaluated. The team members work together to develop staffing estimates for CCSAS CSE project staff for ongoing support once the alliance team completes the two-year timeframe.

Alliance and CCSAS CSE project staff members meet to jointly review roles and responsibilities related to the ongoing system management. Skills and knowledge related to the roles and responsibilities are clarified, as required. Additionally, alliance staff members review pertinent system documentation with CCSAS CSE project staff. Documentation is stored in an agreed upon location and in a logical structure that makes it simple to find.

After the completion of the knowledge transfer process and turnover, CCSAS CSE project staff operate and maintain the application.

3C.7.5 Knowledge Management Curriculum

As described earlier in this section, the alliance's knowledge management approach includes the creation of Individual Development Plans to identify the training requirements of CCSAS CSE project staff. Critical to the development of Individual Development Plans for the CCSAS CSE project staff is the training curriculum.

The alliance has identified formal instructor-led training and web-based training courses that enable CCSAS CSE project staff to develop the skills they need to fulfill the responsibilities of their positions throughout the life of the project. This includes formal training, such as third-party vendor and alliance-developed training.

The alliance has identified training courses that will help develop skills in programming languages, design, development, and testing tools, software operations, and hardware operations required to develop the business solution. The State Training Matrix shown in Figure 3C.7-5 contains course identifications and the positions to which they would apply. Course descriptions are also provided.

The following table summarizes the formal training courses.

	Number of Resources	CCSAS Exec. Project Director Director	Data Architect	Net-Centric Architect	Business Analyst	Application Designer	Data Access Developer	User Interface Developer	Java Developer	Batch Developer	Integration Architect	Testing Lead	Database Designer	Database Administrator	Database Support	System Administrator	Network Architect	Infrastructure Architect	System Mgmt Architect	Security Architect	PC Support
	47	1	2	1	4	4	3	2	5	2	2	2	1	2	2	4	1	3	3	1	2
CCSAS CSE Project Orientation	47	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
PMOffice User Training	47	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
System Development (GRNDS) Workshop	12			x		x		x	x												
Fundamental of Project Management with Iteration	1	x																			
Object-Oriented Analysis and Design Using UML	10			x		x			x												
Version Management for Users	20				x		x	x	x	x	x	x									
Version Management for Administrators	4				x																
Business Modeling with the UML	15				x	x			x	x											
Object Oriented Programming with Java	8						x		x												
Programming Enterprise JavaBeans using Websphere Application Server	5								x												
Rational Unified Process (RUP) Fundamentals	13				x			x	x	x											
Websphere (J2EE) Application Server Development Workshop	1			x																	
XML Programming and Related Technologies	7								x		x										

	Number of Resources	CCSAS Exec. Project Director Director	Data Architect	Net-Centric Architect	Business Analyst	Application Designer	Data Access Developer	User Interface Developer	Java Developer	Batch Developer	Integration Architect	Testing Lead	Database Designer	Database Administrator	Database Support	System Administrator	Network Architect	Infrastructure Architect	System Mgmt Architect	Security Architect	PC Support
	47	1	2	1	4	4	3	2	5	2	2	2	1	2	2	4	1	3	3	1	2
Basics of Automated Testing Tools	2											x									
DB2 Universal Database V6.1 Fundamentals	10		x				x						x	x	x						
DB2 Universal Database V6.1 Administration	5												x	x	x						
DB2 Universal Database Administration Workshop for UNIX	4													x	x						
DB2 Developer Fundamentals	6		x				x						x								
AIX 5L Installation and Maintenance	4															x					
Object Oriented Programming with Java	8						x		x												
WebSphere Application Server V3.5 Administration	7															x			x		
DOORS Administration and Management	6											x				x					
Using DOORS for Requirements Management	14				x	x						x				x					
WebSphere Studio Application Developer	6					x		x													
Tivoli System Administration	3																		x		
Tivoli Security Management Overview	4																		x	x	
IBM pSeries and AIX Overview	4																x	x			

Figure 3C.7-5 CCSAS CSE Knowledge Management Curriculum - *Our knowledge management curriculum provides formal training to facilitate the development of skills and knowledge of CCSAS CSE project staff.*

3C.7.6 Conclusion

Our knowledge management approach provides a structured means for providing training, learning activities and work assignment to facilitate the development of knowledge and skills of CCSAS CSE project staff. Our approach calls for up-front planning for skills and learning needs for each CCSAS CSE project staff member. These plans are assessed and updated on a regular basis. Multiple channels are available to provide learning opportunities. The objective of our approach is to ready CCSAS CSE project staff to effectively operate, support and maintain the CSE system.

3C.8 Technical Reviews

Our approach recognizes the importance of regular checkpoints to assess progress and take corrective action if necessary. Technical reviews are conducted to assess the degree of completion of technical efforts related to major milestones before proceeding with further technical efforts.

3C.8.1 Introduction

This section underscores the importance of technical reviews. The alliance recognizes the scrutiny that the CCSAS project is under and recognizes the value in involving the right people in verification and validation of project processes and deliverables.

Technical reviews are an integral part of many important development and implementation tasks of the CCSAS project. In some cases, they are a capstone to reach confirmation of key deliverables. Our review procedures can also result in other positive benefits as described in Figure 3C.8-1 below.

Review Procedure	Benefit
Ten business day period for reviewing project deliverables	Provides the State with sufficient time to make a thoughtful review of project deliverables before a technical review meeting is conducted.
Incremental review for large deliverables	Breaks up large deliverables into manageable pieces. (Only if necessary, determined during course of project at mutual discretion.)
Setting deliverable expectations	Agree on deliverable content and format up front to reduce time spent on rework.
Formal and informal technical reviews	Provides a venue for resolution of outstanding issues, enabling deliverables to be approved within scheduled timeframes. Enables the State to verify that system requirements are met. Allows the State to monitor the progress of the CCSAS project. Provides the State with an opportunity for increased knowledge transfer.
State Sign-off	Provides for State approval. Allows for scope control and requirements confirmation.

Figure 3C.8-1 Technical Review Procedures and Benefits - *Our technical review approach provides adequate time for review and maintains scope control.*

3C.8.2 Formal Technical Reviews

Formal technical reviews related to major milestones are completed throughout the project life cycle. A diagram correlating technical review occurrence points with the project life cycle can be found in Section 3C.11 System Life Cycle Model (technical reviews are indicated on the Software Development Dependency Diagram). As this diagram shows, the technical reviews form a logical progression through the system engineering process.

The alliance agrees with the following principles regarding formal technical reviews:

- Reviews are a joint effort between CCSAS project staff and alliance members
- CCSAS project staff and alliance members with the appropriate knowledge of the deliverables to be reviewed attend review meetings
- The CCSAS project staff and the alliance co-chair the reviews
- Review decisions made as a result of the review are implemented by the alliance and/or the CCSAS project staff as appropriate
- The alliance monitors that work products and deliverables developed by any subcontractor within the alliance are properly reviewed according to the details provided in this approach
- Technical reviews provide the State with the ability to assess the degree of completion of technical efforts related to major milestones
- The alliance review leader prepares and distributes the technical review agenda and review materials
- Technical review minutes are documented and distributed to meeting participants
- The alliance includes a high-level analysis of the business process re-engineering impact as it relates to the scope of the review
- Technical reviews are conducted in a manner consistent with the process described in the IEEE 1028-1997, Standard for Software Reviews

The following is provided as a list of formal technical reviews to be delivered during the project.

3C.8.2.1 System Requirements Review (SRR)

The purpose of this review is to determine if the system requirements have been captured. Input for this review is the System / Subsystem Specification (SSS) document. The SRR is the decision point to allow the project to proceed with the in-depth analyses and design associated with system design.

3C.8.2.2 System Design Review (SDR)

The purpose of this meeting is to determine if the top-level system design provides an adequate basis for proceeding to Software Requirements Analysis. Input for this review is the System / Subsystem Design Description (SSDD) document. The SDR is the

decision point to proceed with system specification flow-down to hardware and software configuration.

3C.8.2.3 Software Requirement Review (SWRR)

The purpose of this review is to determine if the software requirements have been captured. Input for this review is the Software Requirements Specification (SRS) document.

3C.8.2.4 Software Design Review (SWDR)

The purpose of this review is to determine if the Software Design and Logical Data Model provide an adequate basis for proceeding to Detailed Design. Input for this review is the Software Design Description (SDD) document and the Logical Data Model.

3C.8.2.5 Version 2 System Verification Test Readiness Review (SVTRR)

The purpose of this review is to confirm that system test procedures are complete and to confirm that the alliance is prepared for system verification and performance testing. Input for this review is the System Verification Test Description (SVTD) and System Performance Test Description documents.

3C.8.2.6 Operational Readiness Assessment and Review (ORAR)

The purpose of this meeting is to review the completion status of development prior to the execution of a production go-ahead decision. Input for this review is the System Test Verification Report, the Pre-Implementation Readiness Assessment(s), and the System Qualification Test Report (completed by the State). The risks in exercising a go-ahead decision are reviewed, and a go or no-go decision is reached.

ORARs are conducted at three points in the project – just prior to the rollout of Statewide Services for Version 1, just prior to Pilot rollout for Version 2, and just prior to the rollout of Version 2 software to the remainder of the post-pilot counties. The table below presents the objectives, inputs, and responsibilities related to ORARs.

Software enhancements and “bug fixes” will follow an approach similar to the formal ORAR process. These releases will be addressed outside the formal ORAR process and will follow procedures defined in Configuration/Release Management.

Milestone	Objectives	Inputs	Responsibility
ORAR for v1 Statewide Services			
	Is the software and infrastructure ready for production?	System Verification Test Report	Business Partner
	Does the State agree the s/w is ready?	System Qualification Test Report	State
	Has the Business Partner completed its implementation responsibilities for Statewide Services implementation sites?	Pre-Implementation Readiness Assessment	Business Partner
ORAR for v2 Pilot			
	Is the software and infrastructure ready for production?	System Verification Test Report	Business Partner
	Does the State agree the s/w is ready?	System Qualification Test Report	State
	Has the Business Partner completed its implementation responsibilities for CCSAS v2 Pilot implementation sites?	Pre-Implementation Readiness Assessment	Business Partner
ORAR for v2 Rollout			
	Is the software and infrastructure ready for production?	System Verification Test Report	Business Partner
	Does the State agree the s/w is ready?	System Qualification Test Report	State
	Has the Business Partner completed its implementation responsibilities for CCSAS v2 Cutover #1 implementation sites?	Pre-Implementation Readiness Assessment	Business Partner

3C.8.3 Informal Technical Reviews

Informal reviews occur throughout the development life cycle. The purpose of informal reviews is for CCSAS project staff and alliance team members to help:

- Create and shape the system of the future
- Engage CCSAS project staff in the development of deliverables and work products (also facilitates knowledge management)
- Discover issues and/or problems early
- Work together to monitor progress and identify risks
- Achieve informal day-to-day interaction in order to keep both sides aware and informed

Open, informal communication is paramount to overall team cohesion.

Informal reviews are not considered a formal deliverable. They occur by virtue of teaming together toward a common goal. The overall objective of this approach is to establish mutually effective communication and solid working relationships between State staff and members of the alliance. This is intended to help bolster knowledge transfer, CCSAS CSE design, and decision-making for both parties. Consequently, formal technical reviews tend to run more smoothly with the goal of zero surprises for either side.

These informal reviews are conducted as needed by the alliance. This places emphasis on working towards our common goals and fostering open communication. For example, this informal review approach applies to the system development process as described in the subsections of Attachment 3C.12 System Development. Some examples are:

- Informal review of System/Subsystem Specification (SSS) in 3C.12.1;

- Informal review architecture design along with requirements in 3C.12.2 and 3C.12.4;
- Informal review of Software Requirements Specification (SRS) in 3C.12.3;
- Informal reviews regarding our levels of testing as per the “V” model in 3C.12.9;
- Informal interim design reviews in 3C.12.5;
- Informal code reviews in 3C.12.5; and
- Informal Software development reviews.

Informal technical reviews are a key communications tool to be used throughout the system development lifecycle with the intent of providing continuous feedback to help the CCSAS project achieve its business goals and requirements.

3C.8.4 Technical Review Schedule

The purpose of the technical review schedule is to determine and establish a timetable for reviews. The formal technical review schedule proposed by the alliance is represented in Figure 3C.8-2. The table shows the type of formal review as well as when the review is scheduled to occur during the system development life cycle.

Activity	Type of Review	Schedule
System Design	SRR	Toward the end of system requirements analysis, conducted once per version.
	SDR	Toward the end of system architecture design, conducted once per version.
Software Design	SWRR	Toward the end of software requirements analysis, conducted once per version.
	SWDR	Toward the end of software architecture design, conducted once per version.
Testing	SVTRR	Before System Verification and System Performance Tests are conducted, conducted once per version.
	ORAR	Before a version of the CCSAS CSE system is deployed to production, once for each version.

Figure 3C.8-2 Technical Review Schedule - *Our technical reviews allow for a controlled development process to achieve software design compatibility with performance and business requirements.*

3C.8.5 Technical Review Approach

3C.8.5.1 Process for Technical Reviews

Joint technical reviews are conducted at the alliance's primary CCSAS project office. These reviews are attended by LCSA, DCSS, CCSAS project staff, and alliance personnel with technical knowledge of the processes and item(s) in review. Reviews are also attended by the appropriate State and alliance management personnel to facilitate the decision making process.

Meetings are conducted as per the prepared agenda. The reviews focus on item(s) intended for the specific review. Participants are expected to have reviewed the material before the meeting and be prepared to focus on technical issues not minor items like misspellings.

Minor items are handled outside the framework of the review meetings by supplying the Review Leader with either a softcopy or a hardcopy markup detailing the desired changes. The markups should be delivered to the Review Leader before the meeting is conducted. The Review Leader may bring up markup items during review meeting if such items have a potential effect on project schedule and/or scope.

Technical issues regarding the item under review are discussed and decided upon. Pending items are recorded as action items and assignments and due dates are established for action items before the meeting adjourns. Major issues are recorded and promoted to project issues management process. The resolution of open issues formally concludes the technical review.

3C.8.5.2 Roles and Responsibilities

Figure 3C.8-3 summarizes the roles and responsibilities that are specific to the technical review process. Individual participants may act in more than one role.

Role	Responsibilities
Management staff	• May participate in the technical review for the purpose of identifying issues or problems that require management resolution • Reads the agenda prior to the meeting • Adds the meeting to calendar • Directs correct personnel to attend reviews • Coordinates with review leader

Review leader	• Provides agenda to review participants
	• Invites the appropriate participants
	• Reviews the purpose of the meeting with attendees
	• Defines review meeting specific tasks and roles
	• Sets and/or posts meeting ground rules
	• Follows agenda and keeps meeting on track (topics and time)
	• Drives the review of inputs and/or outputs:
	  • Agenda
	  • Deliverable under review
	  • List action items and their status / ownership
	  • List of issues for management attention
	  • List of recommendations made by review team members

Technical staff	• Gathers relevant material prior to the meeting • Reviews material before attending the review and is ready to discuss and focus on major items. • Completes action items that are due • Participates actively in the evaluation of the item under review

Role	Responsibilities
Recorder / Time Keeper	• Tracks time and informs review leader to move on to next agenda item • Maintains a "parking lot" to capture information that is outside scope of topics listed on the agenda • Documents anomalies, action items, decisions, and recommendations made during the review • Recaps action items at the conclusion of the review

Figure 3C.8-3 Technical Review Roles and Responsibilities – *To conduct effective technical reviews, specific roles are defined for participants.*

3C.8.6 Conclusion

Technical reviews are an important part of our overall CCSAS CSE development life cycle. We believe that it demonstrates our eagerness to work with the State to accomplish the business and technical goals of the CCSAS project. The technical reviews provide value by promoting knowledge transfer and project involvement, communicating status and feedback, and helping to promote overall quality output.

3C.9 Technical Quality Management

The cost of avoiding mistakes is frequently much less than the cost of correcting them. Our proactive technical quality management approach embraces the concept of prevention backed by vigorous inspection.

3C.9.1 Introduction

The alliance intends to work with CCSAS project staff to build quality into its development efforts from the beginning. Our quality enforcement activities provide the framework for validating, monitoring, and controlling technical quality throughout the CCSAS CSE project life cycle. Our approaches and processes include and embrace the following principles and are incorporated throughout our approach:

- Standards, practices, conventions, and metrics
- Controlled tests and reviews
- Collection, maintenance, and retention of documentation
- Trace and cross-reference capability for issue, problem, and risk management
- Integrated problem, issue, and scope management
- Appropriate application of tools, techniques, and methodologies
- Established design and coding techniques

By following these principles, the project team uses efficient and effective processes for performing the work, delivers an application that not only meets requirements, but also is supported by appropriate documentation describing the application and decisions made during the development processes.

3C.9.2 Expecting vs. Inspecting Quality

Quality should be expected and not be considered an afterthought.

Technical quality management should encompass the entire development life cycle and consist of more than just a simple inspection process or a set of testing procedures. The alliance uses the principles of the ExPECT model throughout the life cycle to create specific processes and procedures that can be measured to gauge and enforce quality. While the ExPECT model is one that is used by one of the alliance members, its principles are employed on engagements across each of the alliance partners. (The ExPECT model is not specifically referenced in Section 3A.10 Quality Management. However, the technical management team will implement a quality management program using the ExPECT model and will work with the project-wide Quality

Management team to implement consistency between technical quality management activities and the overall project's quality management program.)

The ExPECT model is a five-stage, customer-centered process that measures progress against clearly defined expectations. The model helps identify processes, measurements, and tracking techniques that will be used to enforce strong quality standards. The technical quality management plan also establishes clearly defined processes for monitoring progress, reporting results, identifying discrepancies, and recommending changes.

The ExPECT Model focuses on five key areas highlighted in Figure 3C.9-1:

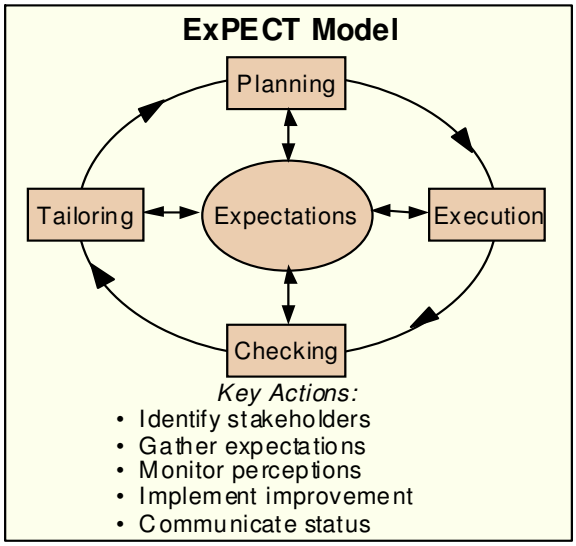


Figure 3C.9-1 ExPECT Model - *Our approach to technical quality management focuses on CCSAS Project Staff expectations early in the process.*

3C.9.2.1 Expectations

A fundamental requirement of our technical quality approach is to understand the expectations of CCSAS Project Staff as they relate to project performance and the quality of deliverables. We meet with staff from across the project to identify expectations coming from different viewpoints. For a project of this scale, different project participants will identify differing expectations. After gathering these expectations, we work with CCSAS project staff to reconcile the information gathered from across the project into a single set of agreed upon expectations. This set of expectations will form the guiding principles for which we can measure success. These documented and mutually approved expectations will function as a primary gauge against which technical quality is measured.

3C.9.2.2 Planning

Planning defines the steps to achieve the documented expectations. We work with the leadership of CCSAS project staff to identify the priority of documented expectations and we design processes to meet those expectations. We create processes to meet the objectives, assign responsibility, define performance goals, and identify metrics and

measurement procedures. We measure team progress in accordance with project milestones. The plan is communicated to the team to verify that personnel across the program understand their role and are prepared to contribute.

3C.9.2.3 Execution

Execution focuses on the implementation of the quality planning as an integral part of client service delivery. Key components of this stage are implementing the detailed processes documented in the plan and collection of the appropriate metrics. Doing so provides the team with an ongoing measure of where it stands with respect to its quality objectives and performance goals. Personnel from across the project are involved in the execution of the plan.

3C.9.2.4 Checking

The goal of this step is to analyze the quality of the project execution and determine improvement recommendations. Checking makes certain that quality objectives are achieved continuously throughout the life of the CCSAS project. Information gathered during the execution phase is used to analyze any gaps between targeted and actual performance goals.

3C.9.2.5 Tailoring

The goal of this step is to continuously improve quality performance. Tailoring focuses on implementing improvements to enforce CCSAS CSE quality standards. CCSAS project staff and alliance members work together to address critical gaps between expectations and performance. Action plans are created and new performance goals and metrics are established to achieve continuous improvement throughout the life of the project.

3C.9.3 Technical Quality Activities Throughout the Development Cycle

As stated earlier, the alliance plans to use ExPECT Model activities to establish quality processes and metrics and to enforce technical quality across the system development activities. Technical quality will be measured in the following activities:

- System requirements analysis
- System architectural design
- Software requirements analysis
- Software architecture design
- Software detailed design
- Database design and development
- Software coding
- System integration
- Test management

- System Implementation (Data Conversion, Change Management, User Training, Hardware and Software Installation)
- Production Support (Operations, Maintenance, Help Desk, Customer Service)

In addition to these system development activities, technical quality will also be measured in the following supporting processes:

- Configuration management
- Requirements management
- Technical risk management
- Knowledge management
- Technical reviews
- Problem resolution

Throughout the project lifecycle, the ExPECT model provides a framework to allow the project team to maintain a proper focus on the "big picture" goals of the project. Rather than only completing individual tasks, the team understands how their work fits into the overall project objectives.

The quality management team works with the application development team to define the overall project expectations and takes periodic checkpoints to measure progress against those expectations. For example, system requirements analysis is the initial opportunity for the alliance to gather and document expectations for the overall functional and technical requirements. By setting these expectations and periodically monitoring compliance against them throughout the project life cycle, we are able to more quickly identify deviations. By identifying deviations early in the process, we are able to realign our work products with the expectations and deliver higher quality into production.

3C.9.3.1 Internal Quality Reviews

The alliance expects formal deliverables to be of high quality before they are submitted for acceptance. Our internal review procedures result in deliverables that meet pre-defined criteria before going to CCSAS project staff for formal approval. Deliverables and work products may be reviewed for quality by:

- The team member responsible for completing the deliverable
- The team member's supervisor
- The CCSAS CSE Subject Matter Expert (SME) working with the team, where needed
- The Team Lead, where needed
- The Quality Management team from the Quality Management office (independent from the application development and other technical teams)

These internal quality reviews improve the processes and products for the project team and shorten review times for CCSAS project staff.

3C.9.3.2 Documenting and Measuring Deliverable Quality

A true measure of performance includes a variety of factors, including quality of the product, whether it is produced on schedule, and whether it meets predefined expectations. We propose measuring performance by implementing a balanced scorecard for technical management. A balanced scorecard tracks major performance indicators to see if they are aligned with the CCSAS project's strategic direction. (Section 2.2.2 Compensation Approach and Performance Measurements provides more detail on the balanced scorecard approach.) ExPECT model principles are used to drive the content and use of the quality scorecard. The alliance has successfully employed balanced scorecards in other engagements, including statewide child support projects. This process enhances the ability to quickly assess how the project is performing in critical areas of the project as it relates to technical management.

3C.9.3.3 Incremental Development Approach

Our incremental development approach builds a functioning and tested subset of a given business capability, one portion at a time. This way, a portion of the application can be demonstrated earlier in the project rather than waiting until the full development cycle is complete. This approach allows the project to assess for quality the key technical components of the system, such as architectural attributes, behavioral requirements, performance requirements, and application functionality. This assessment can occur early in the development process to monitor these quality levels for each subsequent increment. Incremental assessments allow the alliance to further tailor our processes and standards to address quality issues prior to the start of additional development activities. As a consequence, the likelihood of rework is reduced.

3C.9.3.4 Quality Assurance Tools

Quality assurance tools are essential components for achieving a high level of technical quality management throughout the CCSAS project. The tools we currently plan to use include:

- Requirements Management and Traceability:
- Telelogic's DOORS
- Rational Rose
- Configuration Management and Defect Tracking:
- Merant's Product Version Control Software
- Software Performance:
- Sitraka's JProbe

3C.9.3.5 Requirements Management and Traceability

Requirements management provides the alliance with the capability to capture, link, trace, analyze, and manage business requirements. The forward and reverse traceability tool features provide the alliance with a method to determine whether the resulting system matches system requirements.

3C.9.3.6 Configuration Management and Defect Tracking

Our configuration management tool will allow us to organize, manage, and protect the project assets we develop as part of the CCSAS project. This type of tool is essential throughout the course of the development life cycle.

3C.9.3.7 Software Performance

Software performance tools (e.g., Sitraka's Jprobe) allow the alliance to tune software performance to the acceptable level of performance quality.

3C.9.3.8 Problem Resolution

Measuring problem resolution quality is also a key point in project success. Project management should be well informed regarding the status, causes, outcomes, and trends of problems. Various problem resolution metrics allow project management to properly adapt people, processes, and tools to keep the CCSAS project on track.

3C.9.4 Relationship to the Quality Management Program

Our approach is integrally linked with our comprehensive Quality Management Approach. Moreover, the alliance considers technical quality management to be a subcomponent of our overall project-wide quality management program.

Two full time members of the Quality Management team are responsible for monitoring software quality. This includes not only verification of software products, but also the processes surrounding the system engineering processes (e.g., configuration management, requirements management, etc.) For more detailed information about our overall Quality Management Approach, refer to Section 3A.10 Quality Management.

3C.9.5 Enforcing Independence

3C.9.5.1 Independence Achieved Through Project Organization

The alliance proposes the formation of a technical quality team attached to the independent Quality Management organization. This team will be composed of experienced and qualified personnel, who provide technical quality oversight throughout the life of the CCSAS project.

The technical quality management team consists of personnel not directly responsible for producing technical deliverables. This approach is designed to achieve objectivity and avoid bias in quality assurance related procedures and processes.

As shown in Figure 3C.9-2, implementing quality throughout the project requires teams to commit to quality and to be active participants in quality assurance.

Focus	What do we do to deliver quality?	How do we confirm we are delivering quality?
Individual/Task	• Work to achieve high quality task results • Train and communicate • Identify and resolve day to day issues • Work with users, project staff, and management • Manage expectations	• Review individual work products • Assess individual contribution to CCSAS success • Apply supervisor/coach interaction • Verify team members understanding of their role
Team	• Conduct team status meetings • Review accomplishments • Address issues • Check progress • Review work plan • Review risks and mitigation strategies	• Review status reports • Review budget and schedule variance • Meet with team leaders • Review open issues • Review products against metrics
Multi-Team	• Review service level performance • Review formal deliverables • Communicate project progress • Identify and resolve issues • Update CCSAS Project Staff and users	• Review project level status • Assess milestone progress • Monitor deliverable acceptance
Project Level	• Confirm expectations • Assess trends and tendencies • Report on quality results • Analyze risks and mitigation strategies	• Conduct formal quality reviews • Address CCSAS executive input • Assess satisfaction
Quality Management Team	• Confirm expectations • Assess trends and tendencies • Report on quality results • Analyze risks and mitigation strategies	• Conduct formal quality reviews • Address CCSAS executive input • Assess satisfaction

Figure 3C.9-2 Quality and Teamwork - *Our approach "Builds In Quality" by following the principle that each team lead and each team member is responsible for the quality of their work, services, and deliverables.*

Two members of the quality management team will focus on technical quality management. These team members will possess the skill set needed to assess the quality of the products produced and to assess how well the other team members are following the approaches defined in the Technical Management Approach. So, while the technical quality management team has the knowledge and background to effectively assess technical quality, they report directly to the CCSAS CSE Quality Manager so that independence is achieved.

- The technical quality management team reviews deliverables prior to submission to CCSAS project staff.

Further, deliverables created prior to the system development process (i.e., "Plan" deliverables) will be reviewed for quality.

3C.9.5.2 Quality Audits

The quality management team conducts quality reviews of completed and in progress work products produced by the project teams throughout the development life cycle. Quality reviewing of in-progress deliverables identifies quality and compliance issues early in work product development – thereby reducing possible rework. In addition, the quality management team conducts quality reviews of engineering processes. These reviews verify that project processes are executed as planned. As shown in Figure 3C.9.3, technical quality reviews may be performed on several technical work products:

Quality Review Type	Criteria	Proposed Frequency
Design Review	Verify that designs meet system/software requirements Verify that designs are traceable to requirements Verify that designs conform to project design standards	Prior to formal Technical Reviews (System Design Review, Software Design Reviews)
Source Code	Verify that completed code passes through appropriate testing cycles Verify that code adheres to coding standards Verify that code is consistent with design documentation	Ad hoc basis to verify compliance
Process Review	Verify that a system engineering process is executed as described in a planning deliverable	Ad hoc basis to verify compliance
Business Capability	Verify that a completed business capability satisfies the requirements specified in the Software Requirement Specification (SRS)	After System Test

Figure 3C.9-3 Technical Quality Reviews - *Quality reviews identify issues early and help achieve consistent quality across CCSAS CSE teams and in deliverables.*

3C.9.6 Technical Quality Metrics

The alliance plans to work with the State to develop and finalize technical quality metrics for monitoring technical quality throughout the planning, executing, and operating phases of the CCSAS project.

The development and implementation of appropriate metrics is our primary method of monitoring project processes and procedures for quality compliance. Metrics will identify areas of the project that are successful and areas that might need adjustment. Metrics provide project management with the information necessary to make informed decisions regarding corrective action that might be needed.

3C.9.6.1 Assessing Quality

The alliance plans to develop technical quality metrics early in the CCSAS project. Once appropriately planned, developed, and implemented we can accurately measure progress and assess status.

We believe that we should work closely with CCSAS project staff to set thresholds based on proper planning and empirical information after project launch. Individual CCSAS project teams are responsible for collecting and updating metrics. The quality management team reviews the metric reports and provides feedback to project management as well as to individual project teams.

The use of technical quality assurance metrics reduces subjectivity in the assessment and control of software quality by providing a quantitative basis for making decisions about software quality.

3C.9.6.2 Metrics and Thresholds

Software metrics provide a framework for assessing the software quality throughout the CCSAS CSE project life cycle. The use of metrics provides an objective basis for making this assessment. By defining software metrics and establishing target performance targets, software developers know what is necessary to achieve software quality.

Some examples of metrics could be:

- Number of defects identified during testing
- Number of defects not fixed correctly the first time by stage
- Ratio of defect to test conditions
- Average time to fix a defect for a given cycle

We propose to work with CCSAS project staff to define reasonable metrics for the CCSAS CSE project development. Sample metrics thresholds are provided in Figure 3C.9.4.

Metric	Target Performance	Deficiency Action
Number of defects identified during testing	Defects reduce by at least 25% for each subsequent test cycle	More frequent reviews conducted during system development life cycle.
Number of defects not fixed correctly the first time by stage	No more than 15% of defects	Revisit test acceptance/approval processes for possible modification. Programmer personnel who do not meet target performance are identified for more frequent code reviews.
Ratio of defect to test conditions	No more than 10% defect rate during System Verification testing	Review system test procedures. Enforce more stringent review for future System Verification tests for personnel responsible.
Average time to fix a defect for a given cycle	No more than 20% of software coding / unit test budget for the software item	Review of Problem Resolution processes to most accurately track problems. Programmers repeatedly failing to meet target performance are assigned to less complex functions.

Figure 3C.9-4 Sample Quality Metrics and Thresholds - *Quality metrics and thresholds "set the bar" for which quality is measured.*

3C.9.7 Conclusion

We use the ExPECT Model as our framework for finalizing the details of what must be delivered. Using this model allows the team to actively build in quality from the start. The alliance and CCSAS project staff document expectations regarding deliverables, processes, and outcomes. We plan how we can achieve these expectations, execute project work according to the plan, and regularly monitor the quality of the products and processes. This allows us to assess our progress against the expectations, and tailor the plan as needed.

The ExPECT model provides a rigorous quality framework that allows the alliance to build quality into each area of the project.

3C.10 Problem Resolution

Effective problem management is essential for a successful CCSAS project. Central to a quality problem identification and tracking program is an effective problem logging tool, an experienced management team, and a proven problem management process.

3C.10.1 Introduction

The Problem Resolution process used by the alliance accurately tracks and allows for efficient resolution of application issues.

No system is perfect. Problems can and do occur and the alliance plans to have a robust problem resolution process in place to help capture, evaluate, resolve, and control problems as they are discovered on the CCSAS project.

First and foremost, formal problem management is required to enforce project scope. Change cannot be avoided, but unabated or mismanaged change can lead even a well-designed system to failure. Therefore problems are identified, tracked, and approved before any modifications are made to the CCSAS CSE system. If problems cannot be traced back to baseline business requirements, then they are considered out of scope and are referred to our change request management process, as described in Section 3A, Project Management Approach.

Problem reporting is also a key point in project success. Project management should be well informed regarding the status, causes, outcomes, and trends of problems. Various problem resolution metrics allow project management to properly adapt people, processes, and tools to keep the CCSAS project on track.

Formal problem tracking alone does not create a robust process. Our processes are interwoven to leverage the relationships between problem management, project management, and other technical management areas.

The problem resolution process is vital to delivering quality on the CCSAS project. Therefore, the alliance agrees that we should plan and conduct problem resolution management activities in accordance with our Problem Resolution Management Plan (CDL TM 010).

3C.10.2 Problem Resolution Process

3C.10.2.1 Problem Resolution Overview

Problems are captured when an error or an anomaly is detected within a CCSAS CSE product or activity. Information detailing the problem is entered into our problem management tool for tracking and historical purposes. The following graphic summarizes our proposed problem resolution process. Each box represents a step in the problem resolution process; its corresponding problem status is reflected in the process flow below the process step.

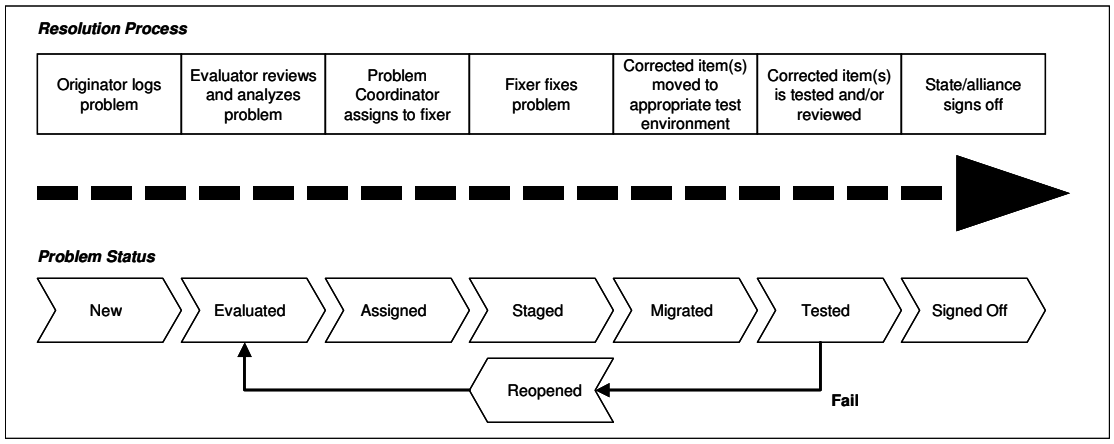


Figure 3C.10-1 Basic Problem Resolution Process - *Our closed loop approach to problem resolution is designed to handle problems throughout CCSAS CSE application development life cycle.*

3C.10.2.1.1 Problem Identification and Capture

Problem logging provides the first step in the documentation process of tracking problems from identification to resolution. Logging starts with the originator (project team member, tester or user) who discovers a problem in the application. The new problem is entered into the problem management tool, containing enough detail to provide problem evaluators a basis to study and diagnose the suspected error. The evaluator then assigns a severity to the problem. The alliance anticipates that severity and date entered will drive the order in which problems are evaluated (i.e., the most severe problems first, with the date entered driving the order within severity). Security features in the product strictly control access to the problem management tool.

3C.10.2.1.2 Problem Evaluation

A problem must first be understood before it can be resolved. The next step in our process is for the appropriate functional and/or technical personnel to evaluate the details of the problem.

Work group team leads are responsible for evaluating problems that arise related to their work area. The problem evaluator begins with a basic review of the problem information

on file. The problem is assigned back to the originating individual if inadequate detail exists to properly evaluate the problem. If enough data is present then evaluators set initial priorities, tracking codes, and dates within the tool. Next they determine whether the problem can be traced back to State approved, baseline CCSAS CSE requirements. If not, then they are marked as an enhancement. If the problem requires a fix then it is marked as such and the evaluators perform an impact analysis.

The impact analysis is performed for the following reasons:

- Scopes the amount of work that is required to fix the problem
- Determines what level of effort it will take to make the change (in hours) and the expected modules that require modification
- Evaluates the impact of the resolution on other areas
- Validates that the resolution does not override or replace a previous resolution
- Reduces chance of duplicate problems

Once the impact analysis is complete and the problem is determined to be within approved CCSAS CSE requirements, functional staff update the tool with the results of the impact analysis along with their recommendations for priority and category. Those problems assigned a higher priority will be scheduled for fixing before those with a lower priority.

The configuration management team uses priority to determine which software build will contain the problem fix. Software releases are scheduled and the project team is made aware of planned fixes contained in the release. Emergency fixes are addressed and are incorporated into the release planning process in accordance with configuration management processes. This controlled approach increases communication to the project team and allows the most critical fixes to be rolled-out first.

If a problem is determined to be an enhancement rather than an application defect (i.e., the requested change is not contained in the system or software requirements), then the problem is considered ‘closed’ and is cross-referenced with the new project scope change request. Project scope change requests are handled by the Change Control Board, as described in Section 3A.4, Change Request Management.

3C.10.2.1.3 Problem Assignment

After problems have been properly evaluated and approved by the problem coordinator, they are assigned to a fixer. Depending on the project phase, a fixer may either be a designated member from the development team (during the initial system development phase) or a dedicated application maintenance resource (during the maintenance and operations phase). As previously stated, problems with a higher priority assignment have precedence over those with lower priority.

Problem coordinators are responsible for assigning problems to fixers, who work to correct the reported problems. It is the problem coordinator's responsibility to estimate the workload of each fixer, evaluate the work break down structure, timelines and objectives, and assign problems accordingly. To verify the problem resolution process is efficient and effective, the problem coordinator should review individual workloads frequently and make adjustments as needed.

Problem coordinators are also responsible for assigning fixers to functional and technical activities, so they may build skills in both areas. Providing cross-training opportunities to the fix team will achieve the following results:

- Reduce the learning curve with regard to functional and technical understanding
- Prevent over-writing previous fixes and recreating old problems
- Provide a point of contact for questions by management about particular functionality (may also help for impact analysis)
- Reduce the chance of introducing new problems when attempting to fix the assigned problem

It is the responsibility of each person on the project to check the status of the configuration items they are working on to determine whether scheduled problem fixes might affect their tasks. Additionally, problem coordinators will contact project personnel that already have affected configuration items checked out in their possession.

3C.10.2.1.4 Problem Correction

Fixers use the problem management tool to locate problems assigned to them and to manage their individual workload by sorting priority and/or severity of assignments. Fixers are responsible for the following activities:

- Re-create the problem
- Determine the specific root cause of the problem
- Inform their supervisor if the work estimate for the problem requires revision
- Physically correct the problem
- Unit test the fix
- Update the tool to reflect problem status and resolution

Problem coordinators track the status of problems and the progress of fixers. Their responsibilities are to:

- Monitor the effectiveness of fixers and coach them as necessary to improve their skills
- Confirm the status of the problems under their supervision
- Validate that the configuration management team is aware of modified configuration items

3C.10.2.1.5 Migration to Test

Our Configuration Management team is responsible for coordinating application migration activities. System administrators complete test environment setup and notify the CCSAS CSE team leaders when setup is complete. Once the environments are set up and initialized, the Migration Coordinator promotes modules and notifies the test team when migration is complete.

After migration takes place, the problem coordinator changes the status of the problem to 'migrated'. Migration from development to the test environment is a scheduled process (with provision for exception procedures for emergency promotion).

3C.10.2.1.6 Regression Testing

The test team is responsible for testing fixes after the appropriate modules have been migrated to the proper test environment. The test team executes test scripts to determine whether the problem is fixed, not fixed, or has introduced more errors.

If a problem fix fails testing, it is marked with a status of 'reopened' and returned to the fixer for another attempt at correction. The fixer will work with their problem coordinator and/or a problem evaluator, as necessary, to complete their assignment. Problem status statistics are a measure of fix team efficiency; therefore, 'reopened' problems are tracked to verify the problem resolution process is functioning as designed.

For more detailed information about our regression testing processes for development modules, refer to Section 3C.12.9 Test Management.

3C.10.2.1.7 Problem Closure and Sign-Off

Problem fixes are considered accepted (signed-off) after the problem has been verifiably fixed and approval is received from appropriate CCSAS project staff and alliance personnel. Problem details contained within the tool are retained for traceability and future reference.

3C.10.2.2 Organizational Entities Involved

Project team members use the problem resolution process to help resolve anomalies found within the CCSAS CSE application. The table below summarizes the role(s) of the project team members in the Problem Resolution Process.

Entity	Interaction
Test team	Executes test scripts and logs problems in the problem management tool; tests/verifies application fixes (regression testing).
Help desk	Logs incidents reported by CCSAS CSE users; attempts to resolve incident with user. Unresolved incidents are promoted

Entity	Interaction
	as 'problems' and are tracked by the problem resolution process.
Problem evaluator and problem coordinator	Evaluates new problems and recommends course of action; manages the overall problem resolution process by assessing priority/severity of reported problems and assigning problems to fix team.
Change Control Board	Sets project priorities and manages scope; approves change requests for application enhancements.
Fixers	Corrects approved problems
Configuration management team	Manages configuration items that are affected by problem fixes
Technical quality team	Monitors statistical information from the problem management tool and reports findings to project management

Figure 3C.10-2 Organizational Entities – *Responsibilities are appropriately shared across the project team to facilitate efficient analysis and resolution.*

3C.10.3 Problem Types

The alliance has identified the following major problem types based on originating source:

- Software problems
- Non-software problems

3C.10.3.1 Software Problems

Software problems are defects in the application that are detected during the development (pre-production) or maintenance and operations (post-production) of the CCSAS CSE system. Our problem resolution process captures and collects software problems after software configuration items have passed unit test. Unit test is considered part of coding and modules cannot be considered defective until after the developer releases them.

3C.10.3.2 Non-Software Problems

Non-software problems are defects that are detected for items unrelated to software configuration (e.g., hardware, documentation). Non-software configuration items might require slightly different handling for evaluation, migration, and testing.

3C.10.3.3 Problem Categories

Problem-types can be further subdivided into categories to enhance reporting and tracking. We propose the following problem categories for the CCSAS project:

Category	Attributes

Category	Attributes
Application	CCSAS CSE system end user receives a system error either generated by the environment or by architecture error handling.
Testing	Tester discovers an error when the test conditions and expected results of a test do not match the actual results. The tester may or may not be given an explicit error, but the failure to meet the expected result must be investigated:
Technical Support	Problem that is more than likely not a software error (e.g., printer does not function). Technical support contacts individual in need of help to attempt resolution of problem.
Training	Problem encountered with training applications or user/training documentation.
System	Operating system or environment based problems.
CSE Program	Problems with the CSE program (e.g., policies or procedures) beyond the scope of the contractor.
Hardware	Problem attributed to hardware failure. Hardware technical support should be dispatched to appropriate location(s) to correct error.
Enhancement	Problem is determined to be out of approved scope. System modifications to accommodate enhancements must be approved by the Change Control Board and be accompanied by a change request.

Figure 3C.10-3 Problem Categories - *Categorization of problems leads to enhanced tracking and reporting capability***.**

3C.10.3.4 Problems vs. Incidents

Technical support incidents are defined as the questions encountered and reported by CCSAS CSE system users who call the Help Desk for assistance. These questions are referred to as 'incidents' to differentiate them from 'problems'. By definition, incidents cannot occur until after the software is released to end-users.

It is expected that the Help Desk will be able to successfully troubleshoot many end-user questions (incidents) before they are logged as problems. If the Help Desk cannot resolve an incident, a problem entry is created in the problem management tool and cross-referenced to the Help Desk incident. At this point, the incident is promoted as a problem and the problem resolution process is initiated. Upon resolution of the problem, the status is updated appropriately in the Help Desk incident log and in the Problem Resolution Management tool. The description of how the Help Desk specifically interacts with the Problem Resolution process will be addressed in the Problem Resolution Management Plan.

For more detail on Help Desk procedures refer to Section 3C.14.3, Help Desk.

3C.10.3.5 Problems vs. Issues

Issues deal with matters of concern at the project level, unlike problems, which are at a lower level and related to specific configuration items of the CCSAS project. Problems can be escalated to the project level issues management process if they cannot be resolved within the problem resolution framework.

For more detail on issue management procedures refer to Section 3A.11, Issue and Action Item Management.

3C.10.4 Problem Resolution Administration

3C.10.4.1 Tools and Techniques

The alliance recommends a problem management software tool that provide the ability to produce pre-defined and ad-hoc statistical reports as well as maintaining backward and forward traceability for the problems – backward to the requirements that are not met due to the problem, and forward to the resolution of the problem (i.e., the configuration items changed in order to resolve the problem). CCSAS project staff access the problem management software tool using their assigned user ids and passwords; security features in the tool control read/write capabilities based on user id.

3C.10.4.2 Reporting

Reporting is entirely dependent on the data that it is drawn from. At a minimum, stored problem information includes a unique problem id, description, category, priority, and status. Data such as this can provide a basis for tracking and reporting.

After fields in our problem management tool are populated we can track, manage, and produce statistical reports about CCSAS CSE development, implementation, and production support related problems and have the data necessary to generate a Problem Resolution Status Report (CDL TM 011).

3C.10.4.3 Analysis and Management

Problem management and resource decisions are based on analysis of the efficiency and effectiveness of the problem resolution process. Analysis is accomplished through the capture of statistical data and trend analysis. Below are some examples of useful statistical data for measuring problem resolution efficiency and effectiveness:

- Aging – Dates are stored for major milestones as problems progress through the resolution lifecycle. One method of analysis could be to calculate the average number of days between milestones. Perhaps resources need to be reallocated within the team to equalize speed across the lifecycle.
- Pipeline – A big picture view of how problems flow through the problem resolution lifecycle could be helpful. This is as simple as taking a count of each problem status code currently in the system. This could identify bottlenecks in the process.

- Stage Containment – It might be useful to examine where errors take place. A count of problems by development stage could indicate whether quality levels require improvement within a given stage.
- Repair Efficiency – A simple count of the problems that were reopened. This could indicate that problems are not fixed properly the first time.
- Outcome Analysis – Maybe another set of counts like the number of problems classified as enhancements, problems closed because they had no basis, or the number of problems deferred or cancelled. A high percentage of non-problems might point to training or help desk issues.
- Priority Counts – A quick count of priorities in the system could uncover the overuse of high or immediate priority categories.
- Configuration Item Counts – If a particular configuration item or groups of items have a higher than normal problem rate it could warrant further investigation into that area.

Statistical data, such as that described above, should be examined for trends. A single statistical data point in time might not provide enough information to take corrective action. But a series of those data points over time could point to a trend that requires action or placement on a watch list. The alliance will work with CCSAS project staff to identify an acceptable set of metrics and the expected value to be derived from them.

3C.10.5 How Resolved Problems Link to Development Activities

Our problem resolution control processes are integrally linked to other CCSAS CSE development activities. These processes are discussed in the following pages.

3C.10.5.1 Configuration Management

Configuration management processes control the migration of the software modules through the environments. The key objectives of our configuration management plan are to coordinate and record decisions, actions and approvals to confirm the following:

- Assessment and evaluation of changes to CCSAS CSE project hardware and software after requirements have been agreed upon and commitments established
- Validating that approved changes are communicated, updated, verified, and implemented properly
- Coordinating the project's day-to-day activities to reduce the risk of conflicting actions on the configuration items involved. This is accomplished through strict and controlled access to code and repositories
- Current status of any given configuration item is readily available

For more detailed information, refer to Section 3C.4 Configuration Management.

The configuration team routinely reviews prioritized problems to determine the impact to the configuration baseline. The configuration team communicates scheduling and/or versioning conflicts to the problem coordinator. As described in our approach, we use a planning process that assigns, a priori, software resolutions to specific builds. In this manner, we prevent version scope creep – that is, incorporating unplanned software fixes into previously planned releases.

3C.10.5.2 Test Management

The Test Management Plan addresses specific tests that are conducted throughout the development of the CCSAS CSE system to verify and validate that the State's requirements are met. The approach, methods, and tools used to plan, conduct, document, and analyze the results for these tests are consistent with our commitment to fulfilling the State's requirements:

- Perform early, up-front test planning, where test conditions are developed as part of specification development
- Execute application and system tests based on realistic business cases with substantial input from CCSAS CSE Subject Matter Experts (SMEs) to help validate application support as well as the business functions are tested
- Develop repeatable test models to facilitate analysis and regression testing
- Follow the industry standard "V" model to confirm test plans are complete and aligned with the design process tailored for the State. The V-model decomposes testing into components that are tightly focused on meeting specific business and functional requirements of the CCSAS CSE system
- Tightly control the technical environment so that application components are tested in a "production-like" environment vs. the development environment(s)
- Employ testing tools that support automated and repeatable testing
- Employ automated tools to cost effectively management test execution

For more detailed information, refer to Section 3C.12.9 Test Management.

The test team performs verification and validation of resolved problems (i.e., regression testing). Problem reports are closed when the test criteria is successfully met.

3C.10.5.3 Technical Reviews

The overall goal of our plan for Technical Reviews is to identify and describe the methods and procedures necessary to satisfactorily complete reviews and the tasks associated with the conduct of each review -- including responsibilities of personnel involved and any necessary procedures (e.g., action item closeout procedures).

For more detailed information, refer to Section 3C.8 Technical Reviews.

During Technical Reviews, problems encountered, problems fixed, and open problems are reviewed. We perform a problem assessment, trend analysis, identify lessons learned, and take action to improve development and test processes.

3C.10.5.4 Technical Quality Management

The Technical Quality Management Plan assesses the CSE system with respect to its functionality, reliability, application, and system performance. It describes technical quality activities, including the types and frequencies of reviews. This Plan details our industry standard approach and processes for validating, monitoring, and controlling technical quality throughout the CCSAS CSE project life cycle. These approaches and processes either make reference to or expound on:

- Standards, practices, conventions, and metrics
- Standard tests and reviews
- Trace and cross-reference capability for issue, problem, and risk management
- Traceable problem reporting, and corrective action
- Appropriate application of tools, techniques, and methodologies
- Development of efficient code

For more detailed information, refer to Section 3C.9 Technical Quality Management Plan.

The number of problems identified, the number of resolved problems, and other metrics discussed in Section 3C.9 are analyzed for trend analysis, threshold settings, and development and test improvement opportunities.

3C.10.5.5 Scope Management

Upon investigation, some testing related problems may prove to be scope changes. Once a problem has been identified as a scope change, it then enters the change request management process for a thorough analysis of the business benefits and identification of the resulting impacts on the project's cost and schedule. This process supports justifiable changes, but defers out-of-scope change requests for future enhancement releases. This approach of integrating problem management and change request management keeps the CCSAS project on course. For more detailed information, refer to Section 3A.4 Change Request Management.

3C.10.6 Conclusion

The alliance brings years of experience in large systems development and the problem resolution process. We have found that effective problem management is essential for progressing in a controlled fashion. We understand how the problem management process interacts with other processes (e.g., test management, configuration management or issue management) Our approach to problem resolution incorporates an effective problem-logging tool, an experienced management team, and a proven problem management process to allow efficient resolution to problems.

3C.11 System Life Cycle Model

Building a large system like CCSAS CSE requires methodology, discipline, know-how, proven principles and experience. For methodology, the alliance proposes the Rational Unified Process. But it is much more than just buying a methodology and using it out of the box; it's about understanding how to use it effectively and how to tailor it. The alliance brings this understanding. Not only does the alliance have experience with RUP on large engagements; it also has additional assets to supplement areas where RUP has shortcomings.

3C.11.1 Introduction

3C.11.1.1 What is a Methodology?

To be truly effective, an enterprise-strength methodology should include many key elements to support development and management of business solutions. It should go beyond software and technology to address:

- Value proposition and business case
- Business process, organization, and training
- Deployment and on-going operations

meth·od·ol·o·gy (mĕth′ə-dŏl′ə-jē)
n. pl. **meth·od·ol·o·gies**
1: a body of methods, rules, and postulates employed by a discipline : a particular procedure or set of procedures
2: the analysis of the principles or procedures of inquiry in a particular field

The methodology should provide proven integrated techniques, guidelines, checklists, and standards as well as classroom training, coaching, and mentoring.

A methodology should provide frameworks for planning and managing the delivery of these tasks and procedures. The planning framework should include work planning templates, estimating models, and phases, milestones, and checkpoints. There should also be elements to track progress, with specific exit criteria for QA processes, and procedures for issues and risk management.

3C.11.1.2 Rational Unified Process

For component based technology development, the Rational Unified Process (RUP) has emerged as a de facto standard. There are three core principles behind RUP:

- **Use-case driven.** That is, functionality is defined from the point of view of the user and business process (rather than a list of features). RUP recommends the use case also be used as the work package for design, implementation, and testing.
- **Architecture-centric.** That is, there is a common approach and model to be used across the system. The architecture specifies the various elements of the solution and how they will interact. For example, it will often specify how business entities will be modeled in the solution, how the application will be layered, and how common problems (such as security and transaction management) will be handled.
- **Iterative and incremental.** That is, key questions and risks are best addressed through prototyping. The solution should be developed in pieces, rather than with a "big bang" approach.

3C.11.1.3 Blended Methodology

RUP has become the market leader for component-based development because of its many strengths. However, we believe that RUP should be supplemented in several areas with the alliance's best practices. RUP places an emphasis on component architecture, with UML-compliant deliverables. It emphasizes iterative and incremental development through robust application design, build and test processes. These processes are well supported by Rational tools and training.

But, RUP's emphasis on iterative development can lead to lack of control if not properly managed. It does not address other application styles such as packaged software, mainframe systems, system maintenance.

The alliance's development methodology has a strong "business capability" orientation. Planning processes and deliverables are critical to success. Disciplined planning techniques include project scope, estimating models, work planning, resource planning, and release planning. Also important are technical risk management and mitigation, configuration management, and technical quality management.

3C.11.1.4 Customized Approach

It is recommended to customize RUP based on a CCSAS CSE specific approach to delivering solutions. RUP, like any methodology, has more deliverables than necessary. While this is intentional and necessary, it is absolutely essential that we customize RUP by using only those deliverables that are important to the success of our project. In this spirit, we would create a representation of the methodology, specific to CCSAS CSE needs. This serves as the entry point for all CCSAS CSE practitioners (i.e., users of the methodology). It would consist of:

- A deliverable flow illustrating only the most important deliverables, as well as deliverable descriptions and templates
- A simplified process flow, including process descriptions, inputs, outputs and participating roles
- Other supporting material, such as an estimating model, training material, etc.
- Links to more detailed information in RUP (e.g., guidelines, tool mentors, etc.)

Our system life cycle reflects this customized approach as described in this approach. The basis of our life cycle model is Rational's Unified Process (RUP) combined with the alliance's methodology and guidance from international standards (i.e., IEEE, ISO). The alliance proposes to use an incremental, component-based development strategy to create California's next generation system. This approach has been used successfully on thousands of engagements in both the public and private sectors.

3C.11.2 Background

3C.11.2.1 The Development Cycle

The RUP life cycle is organized around four phases. In order to avoid confusion with our two-phased strategy we refer to these four life cycle phases as RUP phases. These four RUP phases constitute a development cycle and becomes the foundation of our CCSAS CSE life cycle model.

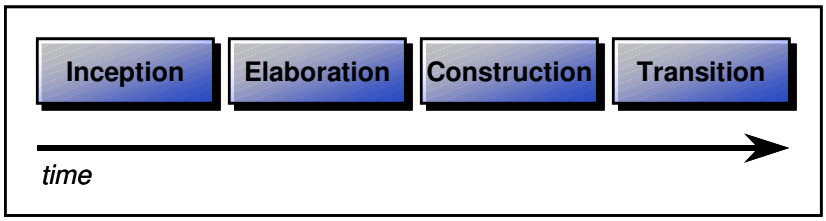


Figure 3C.11-1 The Four Basic Phases of RUP – *RUP's phases lay the foundation of our life cycle model.*

RUP defines its core phases as follows:

Inception phase includes activities like project launch, initial project planning and management, document requirements and set requirements baseline, design and demonstrate architecture, and high-level design.

Elaboration phase focuses on completing higher-level design activities, stabilizing the architecture, and executing key use cases.

Construction phase includes the bulk of detailed design, coding and unit testing, and functional and technical testing of the application before release.

Transition phase involves the activities surrounding the release of the CCSAS CSE application to end-users. This includes conversion, implementation management, change management, hardware/software rollout, end-user training, and on-site support. Production support for the delivered software begins as soon as it is delivered to end-users.

3C.11.2.2 The Principal of Evolution

RUP is structured around continuous improvement. For instance, if a sizeable amount of additional functionality is required then another release of the same application is produced to accommodate the need. Under RUP this is known as evolution.

Evolution repeats the development cycle again, which takes the current generation of the CCSAS CSE application and builds upon it. The content of RUP phases under an evolutionary cycle may differ from the original cycle since a majority of the project level activities, methods, processes, and standards will have been established in the original cycle.

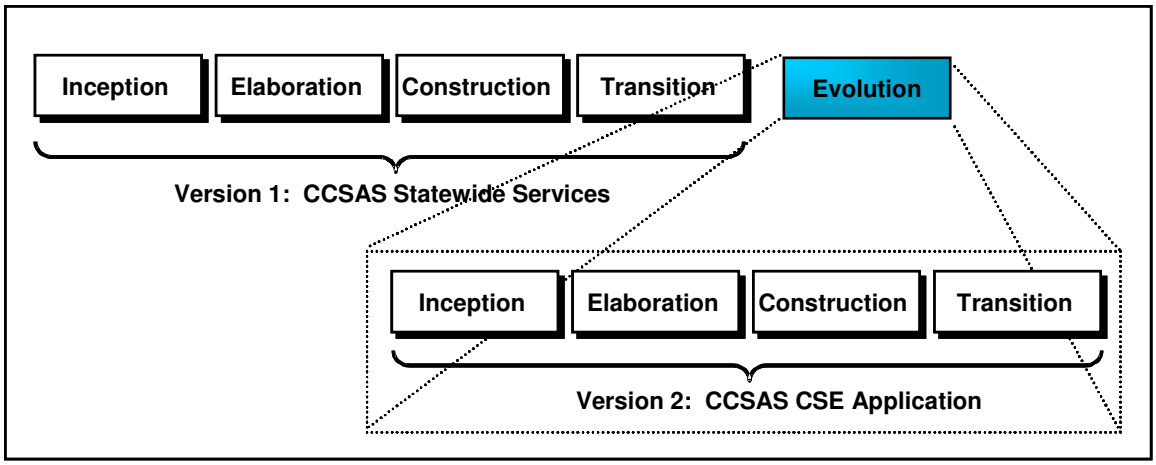


Figure 3C.11-2 RUP's Evolution Phase – *Evolution merely repeats the same four life cycle phases for each new generation of the CCSAS CSE application.*

Figure 3C.11-2 shows how CCSAS Statewide Services starts the evolution process. Evolution plays a key part in our proposed two-phased strategy. Version 1 of our solution creates CCSAS Statewide Services. Version 2 builds upon statewide services. The alliance plans to execute both development cycles in parallel. CCSAS CSE will continue to evolve and have new versions as new federal or state programs may require substantial upgrades to the CCSAS CSE system.

3C.11.2.3 Development Cycle Workflows

There are six core engineering workflows within RUP. Even at a glance these core workflows are easy to understand. Their basic meanings are similar to traditional "waterfall" development life cycle terminology: plan, design, code, test, implement, and run. The following "humpback" diagram depicts Rational's view of core workflow effort over a typical development cycle:

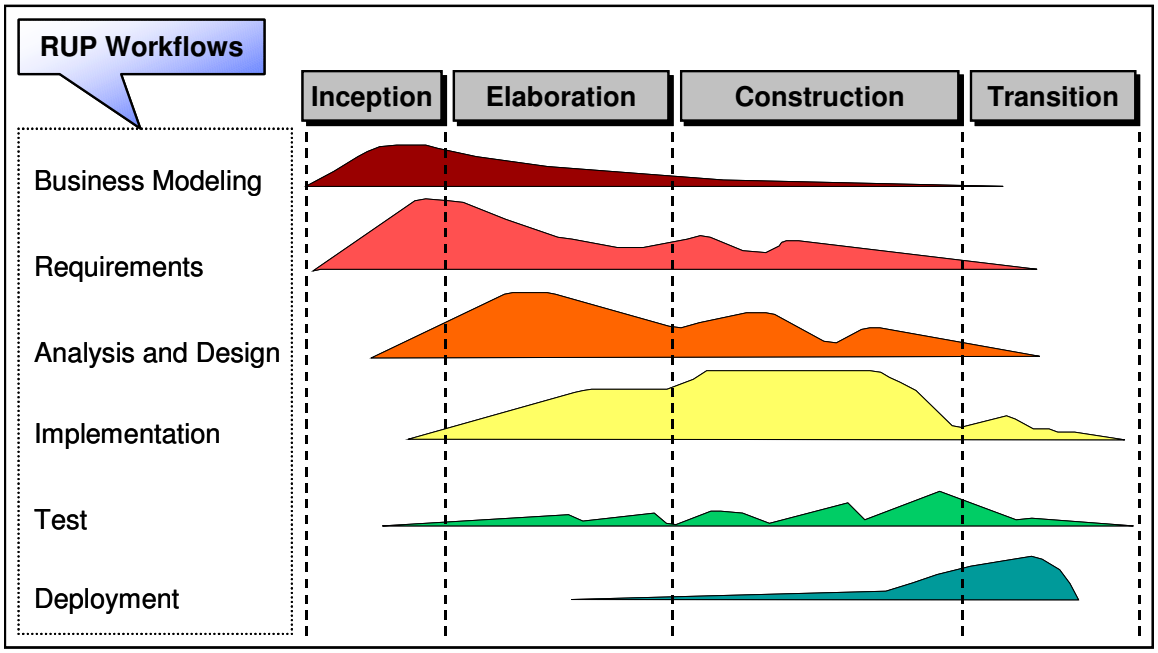


Figure 3C.11-3 The Six Core Engineering Workflows of RUP – *RUP workflows are similar to more traditional life cycle models.*

The core workflows as defined by RUP are:

- Business Modeling - Define the business architecture for the business processes that are in scope. Provide a common understanding.
- Requirements - Reach agreement with users on what the system should do. Plan system test.
- Analysis and Design - Transform requirements into a design of the system. Evolve architecture for the system. Adapt design to match implementation environment. Design for performance. Plan integration test.
- Implementation - Implement classes using components. Test the individual components and classes (i.e., unit test).
- Test - Perform integration, performance, system, and acceptance tests.
- Deployment – Plan and execute the deployment and on-going operation of the system.

Despite its emphasis on more modern techniques, such as component-based development, RUP workflows translate into the terminology of IEEE, and therefore can be mapped directly into the CCSAS CSE life cycle model.

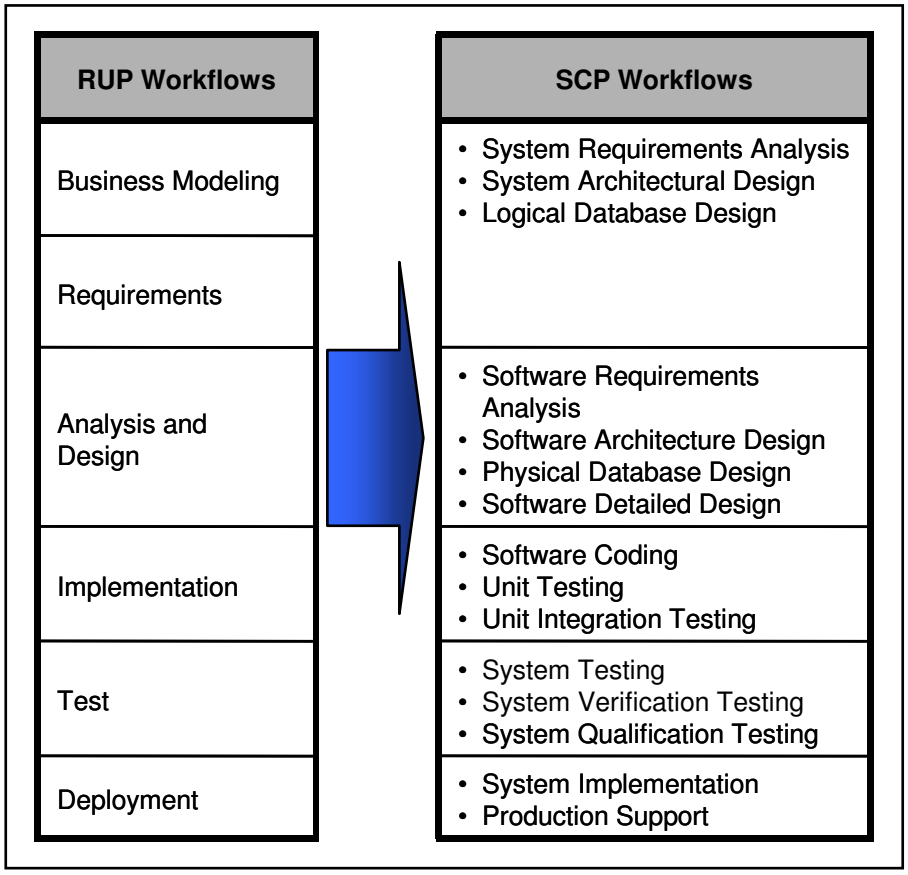


Figure 3C.11-4 RUP Workflows vs. SCP Workflows – *RUP workflows correlate directly to the IEEE life cycle activities.*

RUP is rooted in proven standards, and can be mapped to IEEE. RUP has been used successfully for years by numerous projects, both public and private, to produce quality software applications. RUP provides a solid foundation for our CCSAS CSE life cycle model. The customization of RUP for the CCSAS CSE system development will be documented in the Software Engineering Management Plan (SEMP).

3C.11.2.4 Concept of Incremental Development

RUP is also rooted in the concept of incremental development. Rational's approach emphasizes that incremental development can reduce risk by breaking a large project into smaller and more manageable pieces, or increments.

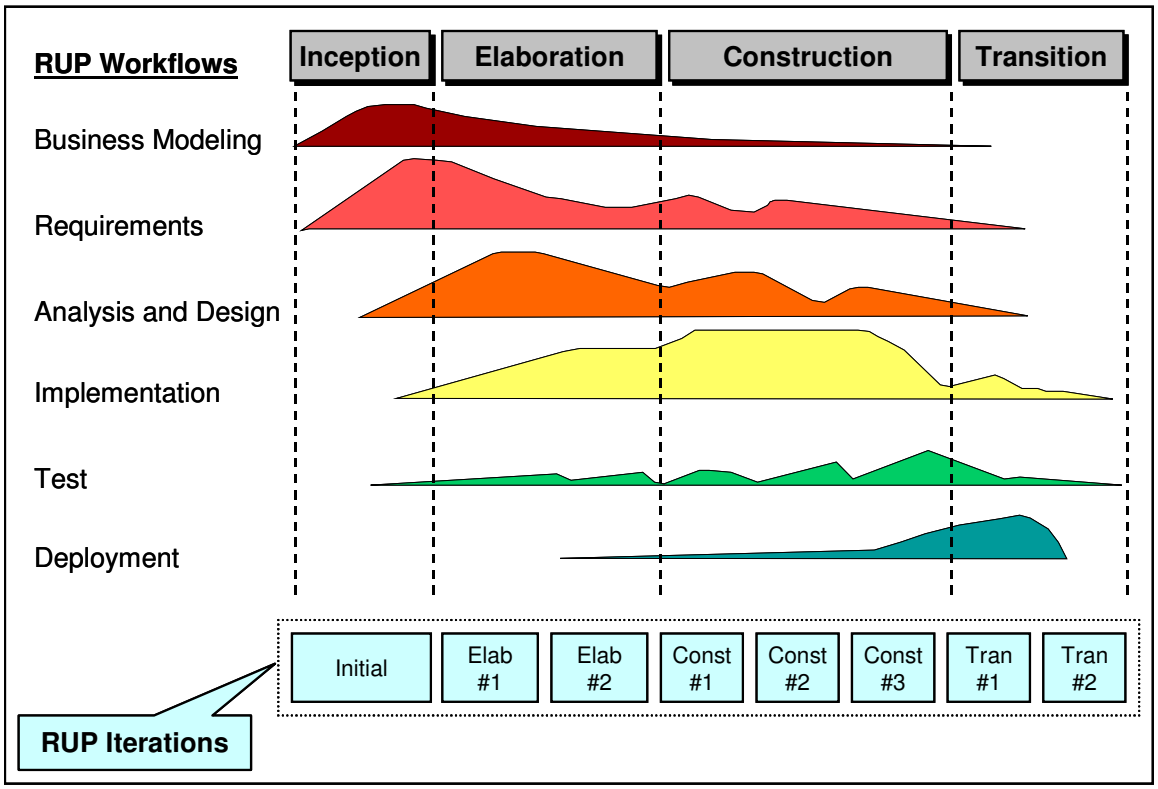


Figure 3C.11-5 The Iterative Nature of RUP – *RUP emphasizes breaking large projects into a succession of smaller iterations.*

The alliance's extensive experience with component-based development has taught us that using increments is a proven concept and that it is also considered by many to be an industry best practice. Therefore incremental development is a cornerstone of our development strategy.

On the other hand, our experience has also shown us that too much iteration can be harmful and that the correct blending of a waterfall approach with iterative elements provides increased management control and higher quality. It is important to define where iterations occur in the project lifecycle to manage requirements and scope and stay on schedule. Otherwise projects can fall into the trap of revisiting requirements and decisions, which lengthen the project. Therefore the alliance proposes a hybrid approach as shown in Figure 3C.11-6.

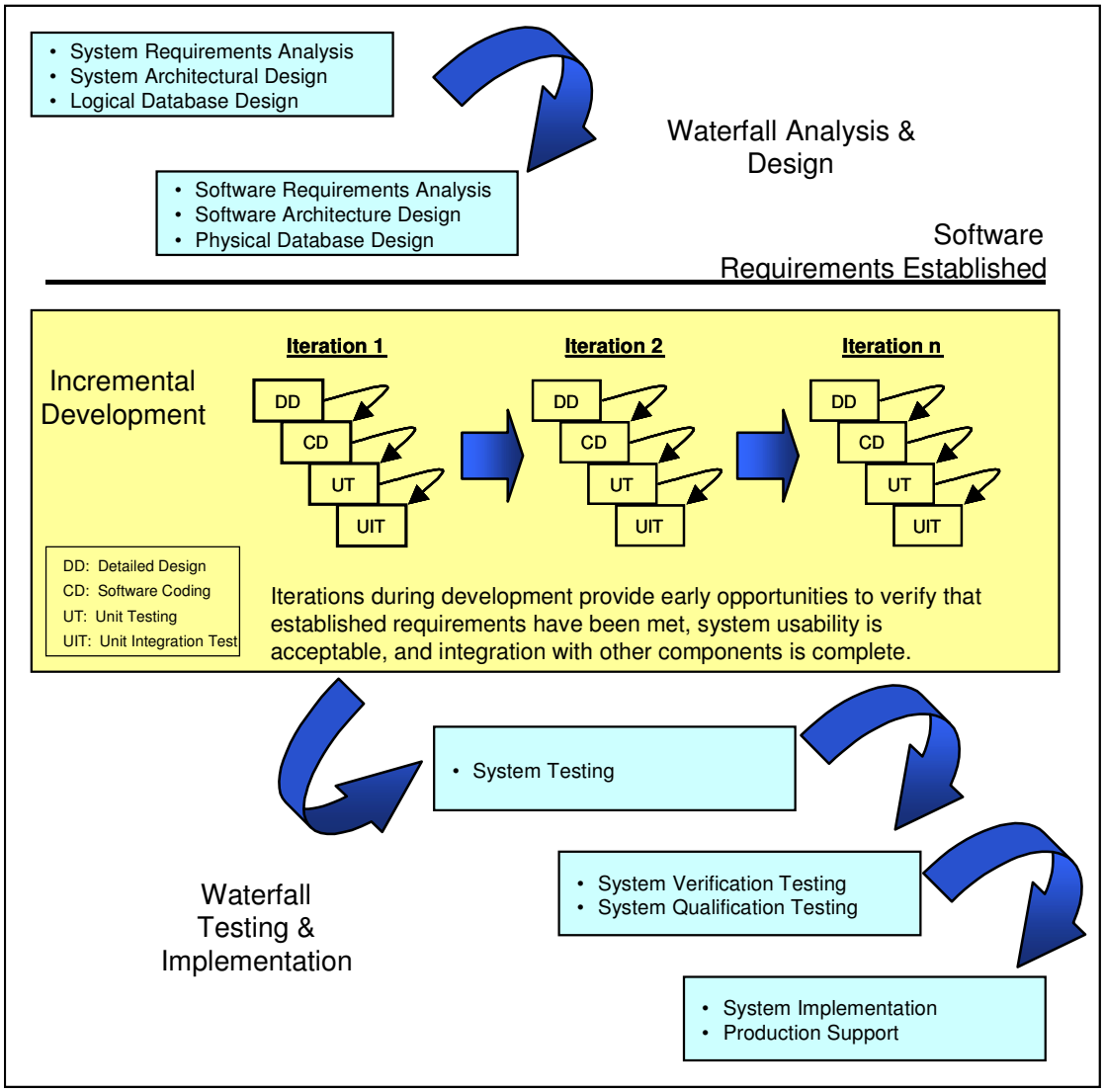


Figure 3C.11-6 The CCSAS CSE Development Process – *Our CCSAS CSE development life cycle emphasizes the proven practice of a combined waterfall–incremental–waterfall approach.*

The alliance proposes a waterfall/incremental development hybrid. We strongly believe that requirements and higher-level design should be firmly established before the majority of coding and testing begins. We have found that it is better to “begin with the end in mind”. Therefore the analysis and design portion of the project follows a more traditional waterfall approach.

Our approach also capitalizes on a form of iterative strategy that we call incremental development. The concept of incremental development centers on taking a set of functionality that has been thoroughly defined during the analysis and design process and dividing the use cases therein into increments. The first increment is then detailed designed, coded, unit tested, and unit integration tested, then the next increment, and so on, until the set of increments is complete. This incremental approach greatly enhances our ability to monitor the progress and effectiveness of the development process. Adjustments can be introduced early versus discovering the issues and/or errors at the end when it is more expensive and cumbersome to fix.

Incorporation of an incremental approach during the development process allows for opportunities to get the product in front of users for feedback early and often. This interaction results in dissemination of knowledge to other project members (Child support business experts, trainers, testers, etc.), refines usability issues, and promotes system user buy in.

Lastly, we return to a waterfall for the remainder of the development cycle so that end users receive a single integrated and tested generation of software.

3C.11.3 CCSAS CSE Life Cycle Model

3C.11.3.1 The Framework

The following figure depicts the fundamental life cycle processes for the CCSAS project.

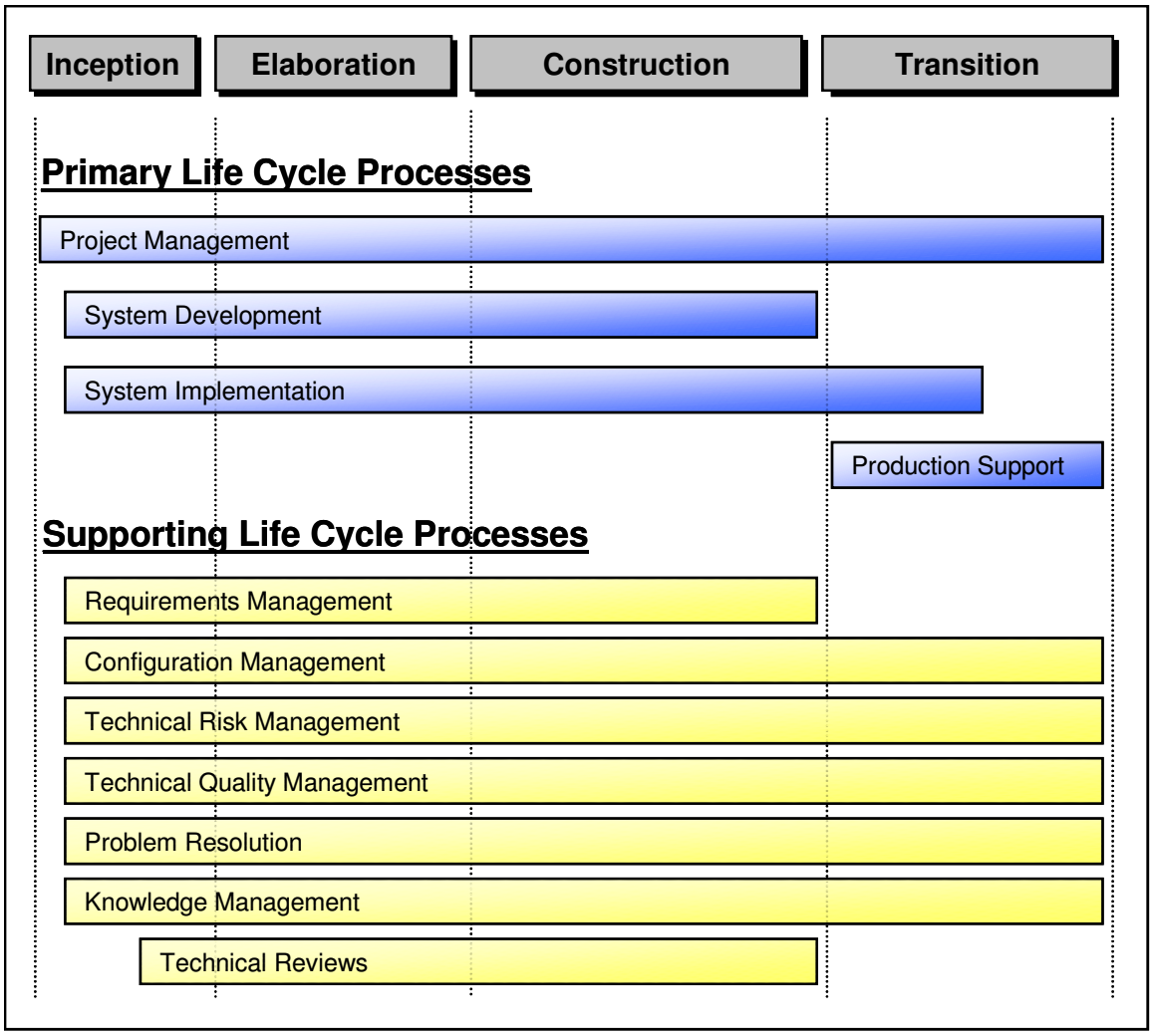


Figure 3C.11-7 High-Level CCSAS CSE Life Cycle Model – *The foundation of our CCSAS CSE life cycle is based on our primary and supporting life cycle processes as described throughout our Technical Management Approach.*

Project Management directs, coordinates, and monitors the activities performed during the other phases to help achieve improved business results. It provides the continuous oversight needed to keep initiatives on track and deliver value. The management process guides the efforts of the State's change program during the entire project life cycle, from inception to realization.

System Development translates the business model into business capabilities. A business capability combines people, business process, and technology elements to collectively improve business performance.

System Implementation prepares the end users and counties to receive the new business capabilities. During this process we work to understand and help refine the overall and county-specific strategies and approaches for achieving the State's objectives. Together, we create detailed plans to help the State and county project teams effectively implement changes and realize and sustain the value the CCSAS project.

Production Support begins when new business capabilities are delivered to end users. The maintenance and operations phase defines how the business capabilities, once delivered, will operate. It focuses on achieving desired performance targets and creating an environment where continuous improvement is the norm. It focuses on maintaining the solutions and delivering the right results.

Supporting Processes provide the needed supporting activities to help manage, develop, deliver, and support business capabilities. Supporting processes help incorporate quality into each step of the life cycle.

3C.11.3.2 The Details

This section pictorially depicts the details behind our CCSAS CSE life cycle model. It includes significant life cycle activities, milestones, and deliverables. This includes technical reviews, quality reviews, delivery of specification and design documents, analysis activities, testing activities, conversion and transition activities, and software deliveries.

Two foldouts follow:

- Software Development Dependency Diagram
- LCSA Implementation Dependency Diagram

Figure 3C. 11-8
Software Development Dependency Diagram Activities

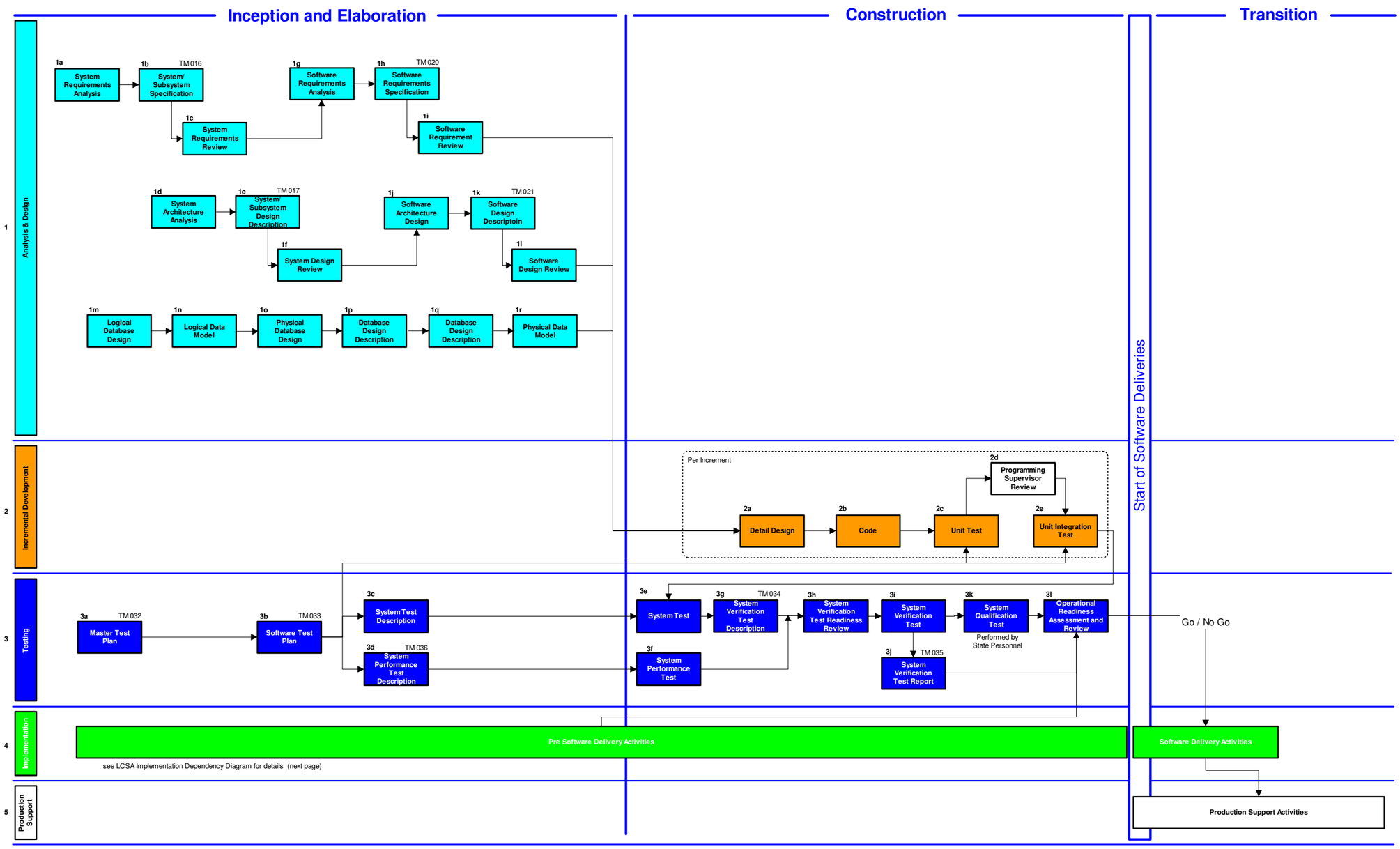
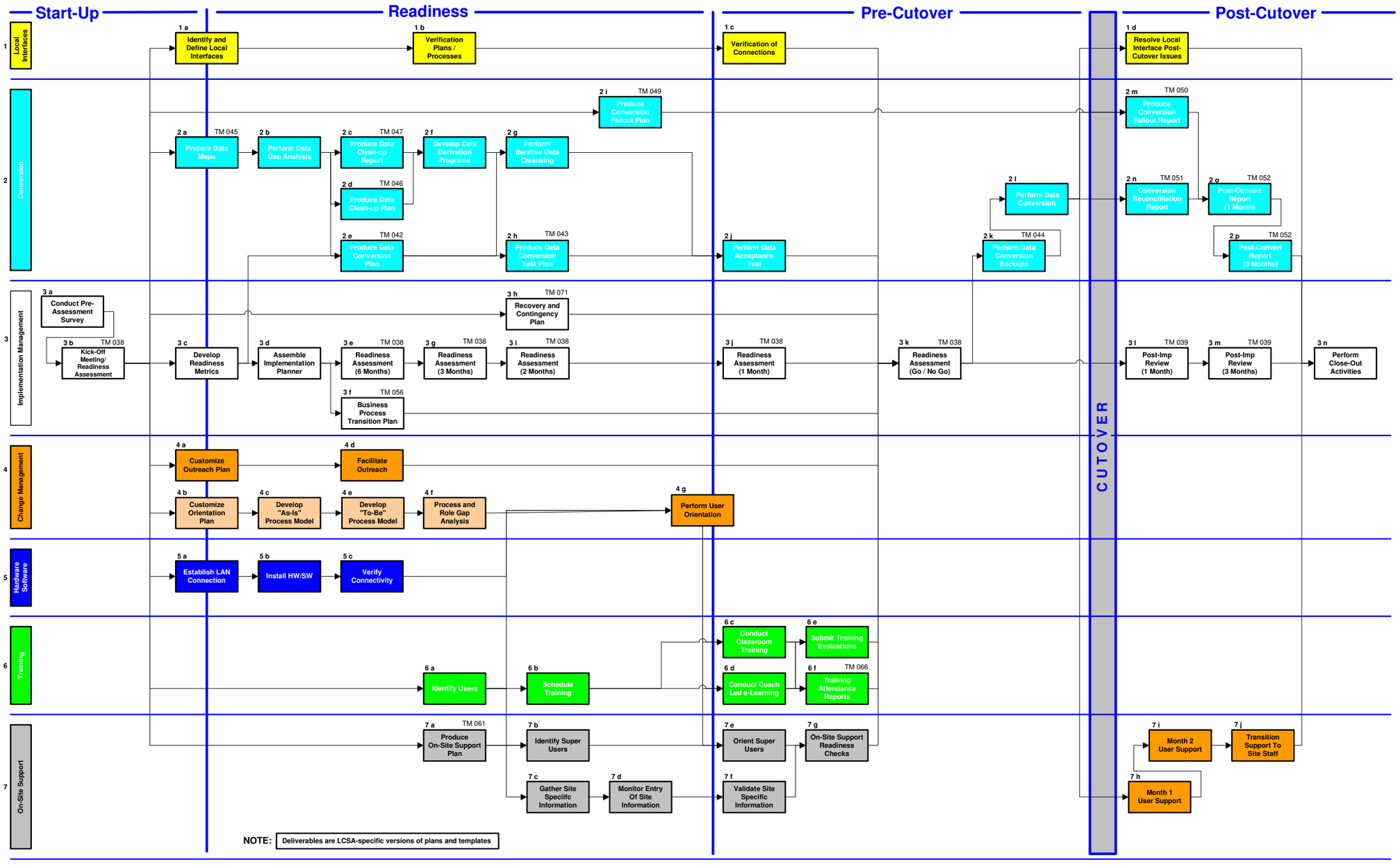


Figure 3C.11-9
LCSA Site Implementation Activities



3C.11.3.3 Core Workflows vs. Business Capabilities

Each version of the CCSAS project contains functionality that is divided into specific business capabilities. Business capabilities represent a subset of CCSAS CSE requirements that share similar functionality traits (e.g., obligation management, adjustments). This allows the alliance to subdivide CCSAS CSE requirements into manageable units of work.

Additionally, similar business capabilities can be grouped into business capability groups. For example obligation management and adjustments would be part of the financials business capability group.

Our core workflows relate directly to phases, business capabilities, and business capability groups. The following concept is integral to our life cycle model strategy, WBS, and project schedule.

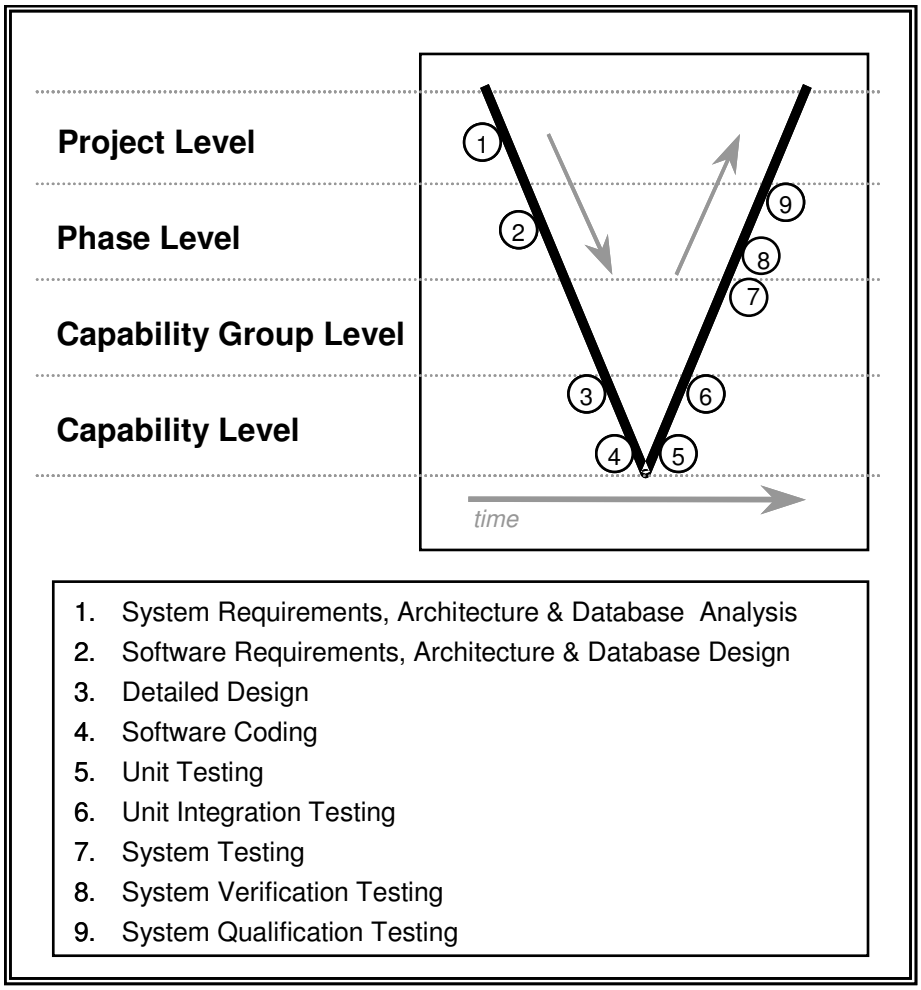


Figure 3C.11-10 Timing of CCSAS CSE Core Workflows – *CCSAS CSE development activities have a direct correlation to child support requirements and functionality.*

3C.11.4 Conclusion

The Rational Unified Process is the market leader for component development and brings many strengths to the marketplace:

- Focus on requirements-based modeling
- UML compliant deliverables
- Robust application design, build and component test processes
- Well supported by tools and training
- Excellent configuration management capabilities

Based on years of applying RUP on many large engagements, the alliance proposes to supplement RUP with their own best practices. The alliance's proposed methodology is a blend of RUP along with our own which includes a strong business capability focus including:

- Business process design
- Training and organization design
- Disciplined planning techniques, project scope, estimating models, work planning, release planning.
- Project tracking and metrics for status reporting
- Technical quality management

It is this combination along with the alliance's knowledge of and compliance with State requirements, industry standards, and proven practices that make us the clear choice to deliver the CCSAS CSE application. Our team provides the right standards, life cycle, and methodologies to deliver the application with a high level of quality.

3C.12.0 System Development

The alliance recognizes the need to focus on requirement consistency, robust designs, architectural attributes, technical requirements, and knowledge management throughout the lifecycle of the project. This overview of our proven approaches and methods shows how we address these key areas.

3C.12.0.1 Introduction

The alliance proposes proven principles, proven work products, proven approaches, and proven tools for system development activities. The sections that follow this document, from System Requirements Analysis (3C.12.1) through Test Management (3C.12.9), discuss our differentiated approach in detail.

A few recurring themes span system development. These themes reappear again and again in the 3C.12 sections. Throughout system development, we consider these themes:

Requirements – the approach for arriving at a comprehensive, consistent, and complete set of requirements[1]

Design – the approach and methods for defining design architectures that satisfy requirements

Architectural Attributes – the approach for incorporating the CCSAS architectural attributes into the requirements baseline

Technical Requirements – the approach for defining and validating system behavioral requirements (i.e., reliability, usability, availability, and fault tolerance) and performance requirements (e.g., speed, latency, and through-put)[2]

Knowledge Management – indication of specific areas where knowledge management is to take place

The alliance agrees that these areas are key to the successful development, implementation, and maintenance of the CCSAS CSE application.

[1] In section 3C.12.0, the word "requirement" refers generically to business, system, and software, unless otherwise specified.

[2] The State defines the term "behavioral requirement" to refer to those requirements related to reliability, usability, availability, and fault tolerance. "Performance requirement" refers to those requirements related to speed, response time, latency, and through-put. Section 3C.12 introduces the term "technical requirement" to refer to both "behavioral" and "performance" requirements.

Because these themes are not unique to a single stage of system development, we address them here, in this overview document. For a complete picture, the reader should consult this document and then find details in the applicable system development sections. For example, this document highlights key messages for knowledge management, across system development. But for the specifics on knowledge management during a stage like System Requirements Analysis, the reader should refer to that particular section. Alternatively, some readers may prefer to first read the 3C.12 sections, and then read this overview when they understand the details.

3C.12.0.2 Requirements

Requirements are among the first items collected and documented during project work. Requirements are the basis for structuring work and are a driver of CCSAS CSE project scope. The alliance intends to work with staff from the CCSAS project, DCSS, and LCSAs to define a comprehensive, consistent, and complete set of requirements.

Our approach for documenting and validating requirements includes the use of automated tools, such as DOORS for requirements management and Rational Rose for component and data modeling. The tools we use are market leaders. However, a tool is only as good as the discipline and experience to use it effectively. Our established design and development approach is founded on the assumption that project tasks and work products are derived directly from documented requirements and can be traced back to them. For more about requirements management, see Section 3C.5 Requirements Management.

No matter how thorough our mutual efforts may be to define requirements, inevitably the need for updates and changes will arise. Managing requirements is one of the most challenging issues that large projects face. Therefore, our approach not only addresses the initial analysis of CCSAS CSE requirements, but also the change process for evolving requirements over a multi-year period. Section 3A.4 Change Request Management discusses the process for addressing enhancements and updates to the requirements baseline.

The alliance recognizes that traceability across requirements and work products is important to the State. As such, we have selected tools, work products, and approaches that support traceability. The individual sections on system development stages, after this document, discuss traceability in more detail.

The following table highlights how we address requirements throughout system development.

Stage	Approach to Requirements
System Requirements (Section 3C.12.1)	System requirements are gathered from a variety of sources. The CCSAS project staff has already completed a substantial effort in gathering requirements. The output of this effort, the Business Requirements, defined the baseline for the CCSAS CSE project. We hold meetings to complete the analysis of these requirements. The alliance brings extensive experience in performing thorough System Requirements analysis. Key principles that guide us in this analysis include: • Use a specialized team trained in requirements analysis • Categorize requirements • Analyze functional requirements with the Rational Unified Process and a use case-driven approach • Incorporate Business Solution use cases • Validate requirements using the System Architecture Design Using the system design for validation helps us arrive at a complete and consistent set of requirements. Similarly, using testability as a litmus test for detailed requirements (as described in section 3C.12.1) helps us arrive at a comprehensive set of requirements. We baseline requirements at two points: Business Requirements are baselined at contract award, and System Requirements are baselined at System/Subsystem Specification (SSS) acceptance. We track requirements in the DOORS requirements management tool. Any new or changed requirements are addressed using the process outlined in Section 3A.4 Change Request Management.
Software Requirements (Section 3C.12.3)	Many of the same approaches and methods from System Requirements also apply to Software Requirements. During Software Requirements Analysis, we examine detailed functional requirements. <u>What</u> the software does and <u>how</u> users interact with it becomes the focus. We again validate requirements, this time with the software architecture. That helps us arrive at a complete and consistent set of requirements. We continue to use testability to arrive at a comprehensive set of requirements. We baseline software requirements at the conclusion of Software Requirements Analysis, after completing the Software Requirements Specification (SRS). We continue tracking requirements in DOORS. Any new or changed requirements are addressed using the process outlined in Section 3A.4 Change Request Management.

Stage	Approach to Requirements
Software Detailed Design (Section 3C.12.5)	We develop Software Detailed Designs based on the software design and user interface standards developed during Software Design. These inputs to the detailed design effort embody functional and technical requirements established in previous phases. Prior to starting a detailed design, the technical architect, software design analyst, and software programmer meet for a kick-off session to discuss the design in context of the functional and technical requirements. An input to this discussion is an extraction from DOORS of the requirements applicable to the design. In addition to the design kick-off session, the alliance employs peer, supervisory, and technical reviews throughout software detailed design. Because these reviews assess the design in context of the software requirements, issues with existing requirements or the need for new requirements may be uncovered. New or changed requirements will be referred to the change control board if they involve scope control. See Sections 3C.12.5 Requirements Management and 3A.4 Change Request Management for details on these processes. By reviewing designs in context of the requirements – and by adhering to established procedures to modify or raise new requirements – the alliance maintains a comprehensive, consistent, and complete set of requirements.
Database Design & Development (Section 3C.12.6)	Requirements for database design are both functional and technical in nature. For example, the federal certification requirements state that the system shall track the following for a child on a IV-A referral: Name, DOB, SSN, paternity establishment, and health insurance coverage information. Case construct flexibility is a requirement. These types of functional requirements impact database design. Database technical requirements are primarily related to performance and security. Our approach for gathering database requirements is similar to that of System and Software requirements. As with other requirements, database requirements are primarily gathered from three sources: the Business Requirements List, system requirements and design, and software requirements and design. Data architects and database designers participate in functional requirements meetings held during system and software analysis and design. As stated in previous sections, we track these requirements in DOORS. An important tool used in database design, development, and implementation is the enterprise-wide data dictionary. This repository includes the name(s) of the data element, a description of the data element, the attributes of the data element, and the reason for including the data element. This comprehensive tool is used project-wide for describing not only the design but also the rationale behind the design. This rationale is embodied in requirements.
Software Coding (Section 3C.12.7)	Software Coding focuses on building out a system based on the set of requirements embodied within the detailed design specifications. The alliance conducts peer and supervisory code reviews to verify that code is consistent with detailed design specifications and the underlying requirements. We address any new requirements uncovered during Software Coding using the change request management process.
System Integration (Section 3C.12.8)	Under the alliance's approach, system integration follows the same overall approach as general system building. However, requirements analysis for System Integration differs from other requirements analysis because there is more involvement from external systems representatives. System Integration requirements discussions focus on limitations and expectations that one system has when communicating with CCSAS CSE. These limitations drive the data transformation logic that is required of our Enterprise Application Integration (eAI) solution (see Section 3C.12.8 System Integration for more detail on eAI).

Figure 3C.12.0-1 Requirements - *The alliance uses proven methods and approaches to arrive at a comprehensive, consistent, and complete set of requirements.*

3C.12.0.3 Design

The alliance recognizes the importance of solid designs. Our experience has shown that incomplete and insufficiently detailed designs may lead to substantial rework during the testing process. And it may lead to code that follows the designs but does not satisfy the requirements. The alliance proposes the use of proven methods and tools to avoid this situation.

Our approach to system design is incremental. It is based on the Rational Unified Process (RUP). See Section 3C.11 System Life Cycle Model for more about RUP fundamentals. Each step in design is based on the requirements that were outlined before it. Each step delves deeper into the details of how the system should be built to support the documented requirements. We verify and validate our designs during review sessions that involve representatives with various perspectives, including database, technical, functional and external.

The following table highlights our approach for design architectures throughout system development.

Stage	Approach to Design
System Design (Section 3C.12.2)	System Design identifies the hardware items, network items, software items, and manual operations necessary to meet functional and technical requirements. The alliance brings a great deal of experience designing and developing robust, flexible systems that implement tried and tested system architecture principles. This includes, but is not limited to: • Integrated View of Development, Execution, and Operations • Architecture Layering • Concentration on Technical Requirements • "Best of Breed" Design Components • High-level, Component-based Design • e-Business Architecture Style • Integrated Data Model • Validation of requirements using the System Architecture Design In addition to hardware and network design, the System Architecture Design includes a high-level functional architecture, which identifies large-grained software items. These functions are designed with Use Cases, which are driven from requirements. These high-level Use Cases provide a starting place for the subsequent Software Requirements Analysis and Software Architecture Design.

Stage	Approach to Design
Software Design (Section 3C.12.4)	During Software Design, we focus on defining how the software addresses its assigned requirements. In addition, the application starts to take shape as we take documented user interface requirements to the next level of detail and we develop prototypes for potential software users. The users interact with these prototypes to validate the accuracy and feasibility of requirements. We continue to use principles addressed in System Design. In addition, for Software Architecture Design, we introduce the following new principles: • Component-Based Design • Proven Practices for Component Architecture • Reusable Architecture Assets for Improving Maintainability The high-level use cases developed during System Design are the basis for more detailed use cases and use case realization diagrams. These visual tools, as well as the prototype, help system users gain confidence in the completeness of their requirements.
Software Detailed Design (Section 3C.12.5)	We develop Software Detailed Designs based on the software design and user interface standards. These inputs to the detailed design effort embody functional and technical requirements established in previous phases. Prior to starting a detailed design, the technical architect, software design analyst, and software programmer discuss the design in context of the functional and technical requirements. Inputs to this discussion are the requirements applicable to the design, embodied in use cases. In addition to holding kick off meetings prior to detailed design, the alliance uses peer, supervisory, and technical reviews during and upon completion of software detailed designs. These reviews assess the design in context of software requirements. By clarifying the requirements applicable to the design upfront and reviewing designs in context of the requirements upon completion, the alliance defines designs that satisfy requirements.
Database Design & Development (Section 3C.12.6)	The alliance uses the data requirements that are gathered during system and software requirements analysis to develop the CCSAS CSE system logical and physical data models. These models along with the database design description undergo multiple peer, supervisory, and technical reviews throughout database design and development. These reviews, conducted by application designers, functional experts, and technical architects, verify that the database design is consistent with the functional and technical requirements defined during system and software requirements analysis.
Software Coding (Section 3C.12.7)	The alliance does not expect to define design architectures during the Software Coding stage of software development. Instead, Software Coding focuses on building out a system based on the set of requirements embodied within the detailed design specifications. As described in Section 3C.12.7, the alliance conducts peer and supervisory code reviews to verify that code is consistent with detailed designs and properly uses application architecture services. These reviews also focus on adherence to project coding and user interface standards.

Stage	Approach to Design
System Integration (Section 3C.12.8)	Under the alliance's approach, System Integration follows the same overall approach as general system building. The rigor and processes applied to non-integration components are applied to integration components as well. A traditional approach to System Integration focuses on message formats and technical requirements that affect a particular message interchange. This traditional approach leaves data validation and transformation logic to the sending and receiving applications. As described in Section 3C.12.8, the alliance uses a different approach – one that is proven, based on well-established layering principles. With this approach, we leverage the capabilities of an eAI software package to centralize and build out message formatting, validation, transformation, and routing logic. Following a shared methodology for both system integration and application development provides ongoing touch points between these two efforts. This encourages proper interfacing between CCSAS application components and the outside world.

Figure 3C.12.0-2 Design - *The alliance proposes proven approaches and methods for system design that focuses on meeting requirements.*

3C.12.0.4 Architectural Attributes

Architectural attributes shape the technical requirements, which in turn shape the design architectures. We use the State's architectural principles and the State's best practices as a starting point. We then add in several proven principles and techniques of our own, based on years of developing maintainable and high-performance Internet, component-based, and eAI systems.

The following table highlights our approach for architectural attributes throughout system development.

Stage	Approach for Architectural Attributes
System Requirements (Section 3C.12.1)	During System Requirements analysis, the alliance reviews, refines, and defines the requirements of the CCSAS CSE system. In performing these activities, the requirements team references guiding principles based on two sources of information: 1) the CCSAS architectural principles and best practices and 2) the alliance's own proven architectural principles. These guiding principles help to create requirements consistent with the objectives of the CCSAS project. As such, CCSAS architectural attributes are incorporated into the system requirement baseline.
Software Requirements (Section 3C.12.3)	As with system requirements, CCSAS architectural principles and best practices guide the reviewing, refining, and defining software requirements. As such, CCSAS architectural attributes are incorporated into the software requirements baseline through their influence on defined software requirements.

Stage	Approach for Architectural Attributes
Database Design & Development (Section 3C.12.6)	The alliance uses the data requirements that are gathered during system and software requirements analysis to develop the CCSAS CSE logical and physical data models. Technical experts participate in formal review sessions, thereby verifying that the models follow agreed-upon architecture principles. These reviews also confirm that specified requirements, including those resulting from architectural attributes, are met. CCSAS architectural principles and best practices influence the design of the database through their impact on the system and software requirements. For example, the need might arise to replicate production data into another data store for reporting purposes. The, the database design team would establish this replicate as read-only as suggested by the CCSAS architectural principles.
Software Coding (Section 3C.12.7)	The alliance does not expect to define new architecture attributes during Software Coding. Instead, Software Coding focuses on building out a system based on the set of requirements embodied within the detailed design specifications. Detailed designs include requirements that are based on desirable architectural attributes. As described in Section 3C.12.7, the alliance conducts peer and supervisory code reviews to verify that code is consistent with detailed designs and properly incorporates the specified architectural attributes.
System Integration (Section 3C.12.8)	Under the alliance's approach, system integration follows the same overall approach as general system building. The rigor and processes applied to building applications apply to integration components as well. This approach incorporates the use of CCSAS architectural principles and best practices specific to integration. These CCSAS architectural principles and best practices, as well as the alliance's own proven practices, guide the definition of requirements and design of interface components.

Figure 3C.12.0-3 CCSAS Architectural Attributes - *The alliance starts with the CCSAS architectural principles and best practices. We add in our own proven principles and techniques, based on years of experience building large-scale Internet, component-based, and eAI systems.*

3C.12.0.5 Technical Requirements

Similar to architectural attributes, technical requirements shape the system and software design. For example, usability requirements are built into user interface standards. Designers and developers then use these standards to create an application with a common look and feel. Technical reviewers rely on these standards to validate that the usability requirements have been satisfied in the software.

The alliance brings a skilled team of architects to fold technical requirements into the CCSAS CSE system. The following table highlights our approach for technical requirements throughout system development.

Stage	Approach for Technical Requirements
System Requirements (Section 3C.12.1)	Section 3C.12.1 System Requirements Analysis describes our proven principles and process for defining and validating requirements. This approach is not limited to functional requirements, but extends to technical requirements. When system requirements analysis begins, the alliance and CCSAS project staff identify decision makers. With the assistance of these decision makers, the requirements team reviews, refines, and defines performance and behavioral requirements. We load these requirements into DOORS.

Stage	Approach for Technical Requirements
System Design (Section 3C.12.2)	System-wide decisions made during System Design are driven by and also impact technical requirements. As such, the alliance proposes that system analysis and design overlap. In doing so, we check the feasibility of performance requirements and resolve technical issues early in the development life cycle. To determine the feasibility of technical requirements, the alliance may perform prototypes to test the design of components fundamental to the system architecture. Where necessary, these prototypes entail the development of simple, incomplete applications that mimic the system architecture. We then test performance and reliability, to validate our approach against the technical requirements. As with performance requirements, we also prototype behavioral requirements for usability. We develop a low fidelity prototype of sample user interfaces during System Design to begin refining and validating usability requirements. System designs undergo peer, supervisor, and technical reviews. Technical architects and database administrators participate in system design reviews to verify design consistency with performance and behavior requirements.
Software Requirements (Section 3C.12.3)	Software Requirements uses low fidelity user interfaces to clarify user expectations, refine usability requirements, and validate usability requirements prior to beginning software design. The alliance develops user interface standards based on the usability requirements refined during low fidelity user testing. By developing this standard before commencing software design, we consider usability and build it into designs from the start.
Software Design (Section 3C.12.4)	We do not expect to define additional technical requirements during Software Design. (If we do, we address them with our standard requirements management processes.) Rather, the alliance focuses on the application of user interface standards into the high-level design of software components. By applying the user interface standards across the software designs, the project team establishes the “look and feel” of the CCSAS CSE application. As with system design, software designs undergo peer, supervisor, and technical reviews to verify that designs comply with the usability guidelines in the user interface standards.
Software Detailed Design (Section 3C.12.5)	We do not expect to define additional technical requirements during Software Detailed Design. We create detailed designs in the context of the software design and user interface standards developed during the previous stages. Prior to beginning software detailed design of a component, the technical architect, software design analyst, and software programmer discuss the design in context of the functional and technical requirements. As input to this discussion, we review the use cases that apply to the design. During this stage of system development, the alliance conducts peer, supervisory, and technical reviews. Technical architects, trained programming supervisors, and database administrators review the detailed designs with an eye on satisfying technical requirements.
Database Design & Development (Section 3C.12.6)	We structure the database design to meet performance requirements. While these requirements may not greatly impact the logical design, they do guide the physical design process. The database design undergoes several peer, supervisory, and technical reviews as it evolves. These reviews verify the database design’s consistency with performance requirements.

Stage	Approach for Technical Requirements
Software Coding (Section 3C.12.7)	The alliance does not expect to define behavioral and performance requirements at this stage of system development. Rather than defining requirements, software coding focuses on the implementation of detailed designs that embody the system behavioral and performance requirements. As described in Section 3C.12.7, the alliance conducts peer and supervisory code reviews to verify that code is consistent with detailed designs – to verify that it meets the behavioral and performance requirements.
System Integration (Section 3C.12.8)	Under the alliance's approach, system integration follows the same overall approach as general system building. The rigor and processes applied to non-integration components also apply to integration components. This approach incorporates the use of CCSAS performance requirements unique to interfaces. The alliance defines performance requirements related to data transfer as well as the allocation of performance requirements between the SDU and CSE applications.
Testing (Section 3C.12.9)	The alliance conducts performance testing to evaluate system and software consistency with performance requirements. Our approach incorporates multiple levels of formal performance testing. This includes code-level optimization tests using JProbe to volume stress tests using LoadRunner.

Figure 3C.12.0-4 Behavioral and Performance Requirements - *The alliance not only defines behavioral and performance requirements, but it also validates that designs and components embody these requirements.*

3C.12.0.6 Knowledge Management

The alliance has a proven track record in building federally certified child support systems as well as using next generation technologies. We feel that including training throughout system development is critical for tooling CCSAS project staff to overcome this complexity. And this means more than just formal training and reading technology textbooks – this means ongoing mentoring, as we build the system together.

We also recognize that child support is a dynamic business where both state and federal policies dictate ongoing changes to the CCSAS CSE system. At the completion of the CCSAS CSE project, the State will operate, support and maintain the CCSAS CSE application. Without a proper knowledge management program, CCSAS project staff might be left in a frustrating situation where they are unable to keep up with the next wave of child support changes.

The alliance appreciates the importance of developing key skills and knowledge in the CCSAS project staff. We propose a Knowledge Management approach that provides opportunities for relevant training for the individual. (See Section 3C.7, Knowledge Management.) This includes formal training in the programming languages, tools, and procedures that we propose.

Our organization facilitates knowledge management, as well. CCSAS project staff participate and contribute throughout the process of designing, building, and implementing the CCSAS CSE application. The alliance proposes a project organization that is "work cell" based. We organize each work cell around a particular function or

technical aspect of the system. Each cell includes one or more CCSAS project staff members. This integration facilitates informal and on-the-job training. It focuses the tasks, and small work groups build a sense of camaraderie and common purpose. This creates an environment that fosters learning.

The following table highlights our approach to knowledge management throughout system development.

Stage	Approach to Knowledge Management
System Requirements (Section 3C.12.1) Software Requirements (Section 3C.12.3)	Some projects train project staff, who source the requirements, how to use the development tools for recording requirements. The project staff are then left to capture requirements on their own. But project staff, who can gather requirements, may not be well positioned to analyze them. We use an alternative approach. A smaller team, consisting of alliance and CCSAS project staff, interviews users and analyzes requirements. We train team members on the development tools and techniques for eliciting requirements. This specialized team then arranges and facilitates discussions with appropriate personnel. CCSAS project staff include a number of key positions for requirements analysis. This team is a cross-section of project staff including project executives, system architects, business analysts, database designers, network and infrastructure architects, just to name a few. Working side-by-side with alliance team members, this collaborative teaming environment supports informal and on-the-job training. CCSAS project staff have knowledge about LCSA operations, FTB system preferences, and DCSS management goals. As well, the alliance brings functional expertise developed on previous child support engagements. From a knowledge management perspective, this expertise enhances the knowledge and experience of the user community. The alliance recognizes that success in system development begins with requirements analysis; it begins by working closely together to capitalize on the strength of CCSAS project staff and alliance personnel. The alliance brings deep skills and experience in successful requirements analysis. We develop these skills in CCSAS project staff through on-the-job mentoring. We also propose more formal training on: • Writing requirements and requirements meeting facilitation • UML use cases and use case development • Rational Rose, the use case modeling tool • Telelogic DOORS, the requirements management tool • Low Fidelity User Interface Design This formal knowledge management aims to improve the efficiency of the analysis process, as well as improve the quality of and confidence in the requirements.

Stage	Approach to Knowledge Management
System Design (Section 3C.12.2) Software Design (Section 3C.12.4)	The conceptual design approach that the alliance proposes uses high-level use cases during System Design and adds additional detail during the Software Design phase. Use Cases and Use Case Realizations are work products from this process. Work cells created for requirements analysis are retained for completing designs. Allowing teams to continue building on the relationships and common knowledge from analysis promotes efficient completion of comprehensive designs. Informal and on-the-job training continue to be important, as those familiar with RUP and Use Case development share information. We provide training for performing system and software design. This includes classes in use cases, Rational Rose, and the Unified Modeling Language (UML). The designers complete Use Cases and Use Case Realizations using these tools and processes. For software design we also host workshops in using GRNDS, our Java application architecture. (See Section 3C.12.4 Software Architecture Design.)
Software Detailed Design (Section 3C.12.5)	The alliance's innovative "Fast Track" detailed design process relies on collaboration between developers and functional and technical Subject Matter Experts (SMEs). The knowledge transfer from these SMEs to the developers facilitates knowledge management. Detailed designs start with a kick-off meeting where the designer, work cell lead, functional architect, and technical architect review the design in the context of the requirements. The designer also receives confirmation on the design during informal design reviews. One or more CCSAS project staff members are involved in each review. This facilitates the knowledge transfer of system component details. Formal training for designers includes training in the use of Rational Rose, UML, and GRNDS.
Database Design & Development (Section 3C.12.6)	The database design requires continual monitoring, maintenance, and tuning. Having a successful transition of knowledge in this area is critical to the application's long-term success. . An integrated team of database designers and administrators, including alliance and CCSAS project staff, provides for an interactive environment that supports informal and on-the-job training. In addition to informal and on-the-job training, we propose a number of formal training courses for CCSAS project staff on the database team. This includes formal instructor-led training and web-based training courses. The alliance has identified a number of formal training classes such as DB2 Universal Database Fundamentals, DB2 Universal Database Administration, and DB2 UDB Administration Workshop for UNIX.

Stage	Approach to Knowledge Management
Software Coding (Section 3C.12.7)	The alliance proposes a project organization bridges the traditional knowledge gap between designer and developer. We do this by having the same acting as detailed designer and developer. This aids in transfer of functional knowledge and improves the quality of each component. Informal and on-the-job training play an important role in the completion of coding, unit testing, and unit integration testing. We hold informal information sessions, including both functional and technical sessions. Formal training for developers includes instruction in WebSphere, GRNDS, Java, XML, and version control. We also conduct a "Writing High Quality Code" workshop, with mandatory attendance by developers. This half-day, interactive workshop includes: • Presentation and discussion of good coding practices, with examples • A group exercise to code review and improve a sample module • Two-person team exercises to code review a sample module. The workshop instructor gives personalized feedback afterwards.
System Integration (Section 3C.12.8)	As mentioned, System Integration shares a common methodology with other system development activities. As such, knowledge management activities are similar to those highlighted above. For example, system integration team members receive training in writing requirements and Telelogic DOORS. In addition, specialized knowledge management for these team members includes training on the eAI software and related eAI concepts.
Testing (Section 3C.12.9)	The Testing team is responsible for System testing. As with the development team, we organize the testing team into functional work cells. These work cells include CCSAS project staff and alliance members. For transition and support purposes, the work cells also include some personnel who participated in the requirements and high-level design phases. Knowledge transfer of the testing methodology, tools, and procedures is important to the long-term success of the CCSAS CSE system. We use informal and on-the-job training opportunities to communicate testing procedures and review requirements throughout the testing process. Formal training for testers focuses on requirements tracking software, automated testing tools, and version control.

Figure 3C.12.0-5 Knowledge Management - *The alliance proposes a Knowledge Management approach that emphasizes integration of CCSAS project staff into formal training, informal training, and on-the-job mentoring.*

3C.12.0.7 Conclusion

The alliance and CCSAS project staff together improve child support enforcement with a statewide system that:

- Adapts to changes in legislation
- Improves timeliness, accuracy, and caseworker efficiency
- Integrates with other agencies, employers, and financial institutions
- Leverages advanced technologies for longevity and lower total cost of ownership

Our system will have these characteristics because we use proven methodology, tools, and principles. And we consider requirements, design, architectural attributes, behavioral and performance requirements, and knowledge management throughout system development, as introduced in this section. The sections that follow on system development, from System Requirements Analysis through Test Management, describe our approach in more detail. This approach will lead us, in a disciplined way, towards a successful outcome: a CCSAS CSE system that meets the expectations outlined in requirements – and a CCSAS project staff team that is ready to own and maintain that system.

3C.12.1 System Requirements Analysis

We use repeated, detailed analysis to prepare requirements for design – to get them in a form so they are traceable, testable, and feasible. Our proven approach to component-based development begins with System Requirements Analysis, as we draw on the Rational Unified Process, use cases, and IEEE. Lastly, we overlap System Requirements Analysis and System Architecture Design. This not only validates that our design meets the requirements, but it also improves those requirements and drives them out in more detail.

3C.12.1.1 Introduction

Projects often conduct requirements analysis by:

1. Having development team members meet with the system user community;
2. Writing down everything users want in the system; then
3. Beginning design according to those requirements.

While commonly used, this approach rarely yields the information critical to begin system development. The true requirements eventually materialize, but not until much later in the development process. And often too late.

Gathering requirements, as described above, is an important step to understanding a system's purpose. In fact, the State has already performed the gathering step by completing the Business Requirements List. But requirements are not ready for design until we have conducted a thorough analysis.

More than writing down what the user says: thorough analysis is the key to well understood, high quality requirements.

A repeated, detailed analysis clarifies vague requirements, filters out unnecessary ones, and eliminates duplicates. This section describes our approach to performing a structured and detailed System Requirements Analysis.

The refined system requirements that result from this analysis are the primary input for designing the system architecture. As defined by IEEE, "system architecture" defines the top-level hardware components, software components, and manual operations required by the CSE system. Designing this architecture begins the process for defining lower-level software items, which we decompose during software analysis and design. See sections 3C.12.3 Software Requirements Analysis and 3C.12.4 Software Architecture Design, respectively.

3C.12.1.2 Proven Principles

The alliance brings extensive experience in performing thorough System Requirements Analysis. Key principles that guide us in this analysis include:

- Use a specialized team trained in requirements analysis
- Categorize requirements
- Analyze functional requirements with the Rational Unified Process and a use case-driven approach
- Apply Business Solution use cases
- Validate requirements using the System Architecture Design

The following paragraphs describe these principles.

3C.12.1.2.1 Use a Specialized Team Trained in Requirements Analysis

Some projects train project staff, who source the requirements, how to use the development tools for recording requirements. The project staff are then left to capture requirements on their own. But project staff, who can gather requirements, may not be well positioned to *analyze* them.

We use an alternative approach. A smaller team, consisting of alliance and CCSAS project staff, interviews users and analyzes requirements. We train team members on the development tools and techniques for eliciting requirements. This specialized team then arranges and facilitates discussions with appropriate personnel.

Appropriate training on the process and tools is vital.

As well, the alliance brings functional expertise developed on previous child support engagements. This expertise enhances the knowledge and experience of the user community.

3C.12.1.2.2 Categorize Requirements

We identify various types of requirements for a more complete understanding of system needs. This is in accordance with IEEE 12207.2-1997 Section 5.3.2.1. The following paragraphs describe how and why we differentiate between:

- Version 1 and Version 2 system requirements

- Summary and detailed requirements
- Functional and technical requirements

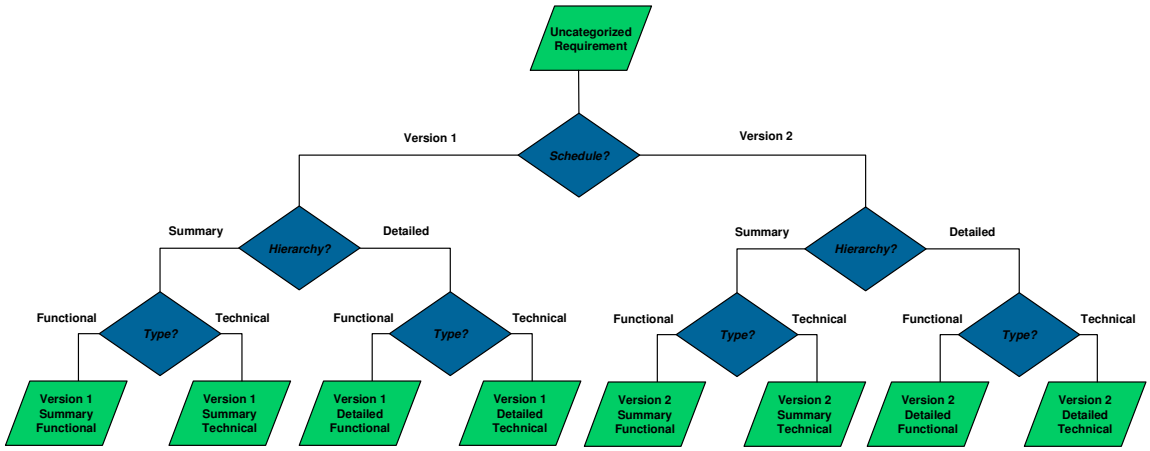


Figure 3C.12.1-1 Requirements Categorization – *We categorize requirements on three independent dimensions, for scheduling and organizing them.*

First we categorize system requirements as belonging to either "Version 1" or "Version 2." Then, we can concurrently categorize the requirements as being either summary or detailed, as well as being either functional or technical.[3]

3C.12.1.2.2.1 Categorize System Requirements as "Version 1", "Version 2", or "Common"

We categorize system requirements as belonging to one of:

- Version 1[4]
- Version 2
- Common (meaning the requirement applies to both Version 1 and Version 2)

We categorize requirements in this manner solely for the purposes of scheduling. Simply, assigning requirements to a particular version determines when we analyze and first address them. Many requirements apply to both Version 1 and Version 2, like those related to hardware. We assign such requirements to a "Common" category. Such requirements are first analyzed along with other Version 1 requirements.

[3] Figure 3C.12.1-1 Requirements Categorization is a conceptual diagram. It does not imply that we categorize requirements as either summary or detailed *before* we categorize requirements as either functional or technical. As noted in the text, we can deal with these categories concurrently.

[4] Section 4.1 Business Solution Narrative addresses the alliance's innovative, phased approach.

Subsection 3C.12.1.5 Proven Process explains how our phased delivery approach affects System Requirements Analysis deliverables.

3C.12.1.2.2.2 Categorize Requirements as "Summary" or "Detailed"

People often describe requirements in a manner that is too broad to be tested. Comments like "The system should respond to a user request in an acceptable amount of time" or "The SDU should disburse payments following accepted methods" are not sufficient requirements by themselves. Instead, we document these high-level statements as placeholders for the more detailed requirements they represent.

In the System/Subsystem Specification (SSS), we denote these high-level requirements as "summary" requirements. Summary requirements cannot be tested. Rather, they act as guidelines for the system designers and builders. Lacking testability, one cannot later validate that these requirements have been properly built. Instead, the summary requirements need to be refined, to be more specific.

In contrast, "detailed" requirements describe particular system functionality or technical capabilities. An example of a detailed requirement is "The system must respond to user requests within an average time of three seconds and not longer than five seconds." This detailed requirement belongs to the more general summary requirement above. During testing activities, we write objective tests to confirm that CCSAS CSE satisfies a detailed requirement.

In addition to improving testability, this two-level hierarchy also helps us create a comprehensive and consistent set of requirements. Any summary requirement without detailed requirements needs additional analysis. Any detailed requirement without a summary requirement has not yet been properly categorized. In both cases, this analysis may occur during System Requirements Analysis or Software Requirements Analysis.

"… should respond in an acceptable amount of time" is not detailed enough – it cannot be tested.

This hierarchical categorization is independent from the version categorization mentioned above. In other words, we have Version 1 summary requirements, Version 1 detailed requirements, Version 2 summary requirements, and Version 2 detailed requirements.

Our use of summary and detailed requirements complies with IEEE 12207.2-1997 Section 5.3.2 Guidance 3 standards. We categorize requirements in this manner to:

- Provide a litmus test for testability
- Organize the fundamentally large space of systems building. With a summary/detailed hierarchy, individuals interested in a particular area can look first for the relevant summary requirements and then zoom in on the details. Otherwise, one would have to wade through a sea of hundreds of requirements.

3C.12.1.2.2.3 Categorize Requirements as "Functional" or "Technical"

We categorize system requirements as either functional or technical:

- **Functional requirements** represent federal and state laws, policies, and regulations; federal automation guidelines; California's legal processes for child support; and automated system objectives related to business and technology goals. In other words, functional requirements describe business processes, business rules, and business data that comprise business logic in the software, as well as in manual procedures.

Summary functional requirements and summary and detailed technical requirements drive system architecture.

- **Technical requirements** represent requirements that relate to system and software quality attributes, management, and infrastructure. They affect the usability, performance, maintenance, and production operation of the system. These requirements impact hardware and system software decisions made in System Architecture Design.

This categorization determines how the team analyzes either type of requirement. During system requirements analysis, summary requirements are the appropriate level for functional requirements. These high-level requirements shape the high-level business component model, which is a fundamental part of the CSE System Architecture. See Section 3C.12.2 System Architecture Design. This component model describes the CSE system's decomposition into well integrated but loosely coupled software items. We look at each of these software items, and their associated, detailed functional requirements, during software analysis and design.

Another way of looking at the distinction is that summary functional requirements drive the system's functional architecture, or "big picture." Detailed functional requirements, on the other hand, pertain to the individual software items that plug into that architecture.

Technical requirements impact system design more than functional requirements do. This is because technical requirements affect system-wide technology decisions related to hardware and software. Therefore, our System Requirements Analysis carefully reviews both summary and detailed technical requirements. We write detailed technical requirements with the "testability" litmus test in mind. This enables testers to later create test conditions that explicitly trace back to their originating detailed requirements.

Categorizing requirements in this manner allows System Requirements Analysis to focus on those requirements that drive the system architecture design.

Moreover, this categorization allows project staff, with different interests, to more easily focus on requirements that are relevant to them. For example, operations support personnel care more about technical requirements, which directly impact the environments, hardware, and system software that they need to support.

This categorization is independent from the version and hierarchy categorizations. See Figure 3C.12.1-1 Requirements Categorization for all possible combinations.

3C.12.1.2.3 Analyze Functional Requirements with Use Cases

Some projects document requirements with simple, numbered text in a spreadsheet or tool. We go beyond this typical approach, by creating additional work products to present different views and perspectives of the functional requirements. We believe that describing the desired functionality of a complex system like CCSAS CSE requires:

- **Descriptions of system interactions.** We analyze interactions with a use case-driven approach. This approach is fundamental to the Requirements Workflow of the Rational Unified Process, which has been proven for large-scale, component-based development.
- **Pictures.** A picture says a thousand words. We believe that these pictures should follow an accepted diagramming standard, allowing reviewers to focus on content rather than diagram notation. We use standardized UML notation for use cases and related diagrams. UML is a de facto industry standard. Section 3C.12.1.4 Proven Work Products describes these diagrams in more detail.

A use case model describes the business goals of a system. Use cases define a system's functional boundaries by decomposing it into manageable pieces and different flows, or scenarios. Use case scenarios represent a single path through that use case. In other words, a use case scenario contains no "if-then" or conditional statements.

Use cases should not describe *how* the system performs its functions. Rather, use cases define system boundaries, which in turn define the scope of the system. This provides a mechanism to validate scope.

A use case "actor" is a role played by someone or something external to the system. Actors can be humans (e.g., caseworker, locate worker, CSE policy analyst) or other systems (e.g., the SDU, TANF systems). Each interaction starts with an initial event from the actor to the system. The use case proceeds through a series of events between the actor, the system, and possibly other actors, until the interaction initiated by the original event reaches a logical conclusion.

A use case is a story with a beginning, middle, and end.

The use case approach provides a structured method for capturing and illustrating the functional requirements to meet CCSAS' business goals. The alliance performs use case modeling during System Requirements Analysis.

3C.12.1.2.4 Apply Business Solution Use Cases

During System Requirements Analysis, we use the high-level use cases already created as part of the Business Solution. See Section 4A.3.2.c Computational Viewpoint. This gives us a significant kick-start for analysis activities.

Inevitably, aspects of this model will change during CSE analysis and design. This is only to be expected for an incremental development effort. In the case of this approach, there is an additional factor: the model was developed without the direct involvement of DCSS or FTB staff. The alliance used the requirements available, its extensive experience, and creative thinking to produce a solution.

During the official analysis and design phases, the combined CCSAS project team evolves and refines the business solution use cases. We view these use cases as a good starting point for the activities described in this section.

3C.12.1.2.5 Validate Requirements Using the System Architecture Design

IEEE 12207.2-1997 Section 5.3.2.2 requires that system requirements be:

- Traceable to the business requirements
- Consistent with the business requirements
- Testable[5]
- Feasible with the system architectural design
- Feasible with planned operation and maintenance of the system

To confirm these points, it is preferable to begin testing the system architecture while analyzing the system requirements. This is because feedback from the design affects the requirements. As such, we consider initial System Architecture Design to be part of System Requirements Analysis. We use the design to help validate that requirements are supportable, and that they truly reflect the project staff and users' intentions.

Overlapping analysis and architecture design improve existing requirements and fleshes out new ones.

3C.12.1.3 Proven Work Products

3C.12.1.3.1 IEEE-Based System/Subsystem Specification (SSS)

The System/Subsystem Specification (SSS) defines the requirements to be met by the system. The CSE SSS is based on Annex F.2.2 of the IEEE J-STD-016-1995 standard. The SSS is a composite deliverable that contains the following:

- **Functional Requirements** – Analyzed summary and initial detailed functional requirements, stored in DOORS
- **Technical Requirements** – Analyzed summary technical requirements and detailed technical requirements, stored in DOORS

[5] The degree of "testability" for System Requirements will be defined in the SSS. "Testable" can be achieved via "Demonstration", "Test", "Inspection", "Analysis", or "Special Qualification" methods.

- **High-Level Use Cases** – The related, high-level use case descriptions and diagrams. These describe a business goal as a series of interactions between actors and CCSAS CSE, and they are stored in Rational Rose

The following paragraphs describe use case diagrams.

3C.12.1.3.2 Use Case Diagrams

The term "use case" actually refers to a composite work product that consists of case diagrams, descriptions, scenarios, and sometimes actor definitions. During System Requirements Analysis, use case diagrams are the most relevant. These diagrams visually depict the interactions between actors and the application, as well as the interactions between different use cases. Actors appear as stick figures, use cases appear as ovals, and the relationships between them appear with arrows. Section 4A.3.2 CSE System Architecture further defines use cases and gives examples.

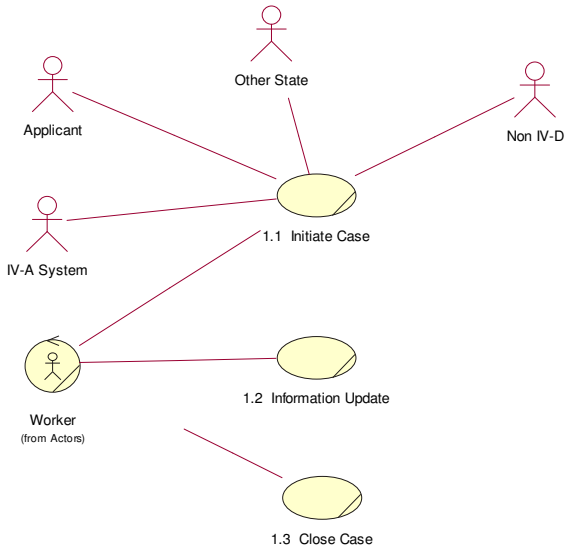


Figure 3C.12.1-2 Example Use Case Diagram - *Case Initiation and Update is defined by three major use cases: Initiate Case, Update Information, and Close Case.*

Use case diagrams capture the essence of the interactions between CCSAS CSE and the outside world. We use Rational Rose, a market leading analysis and design tool, for drawing use case diagrams in UML. We also use configuration management processes, as described in section 3C.4 Configuration Management, for our use cases.

As we proceed with Software Requirements Analysis, we introduce the other artifacts in a use case model. Section 3C.12.3 Software Requirements Analysis describes these artifacts.

3C.12.1.4 Proven Tools

3C.12.1.4.1 DOORS Supports Traceability

We use Telelogic DOORS, a market-leading requirements management tool, for capturing both summary and detailed requirements.

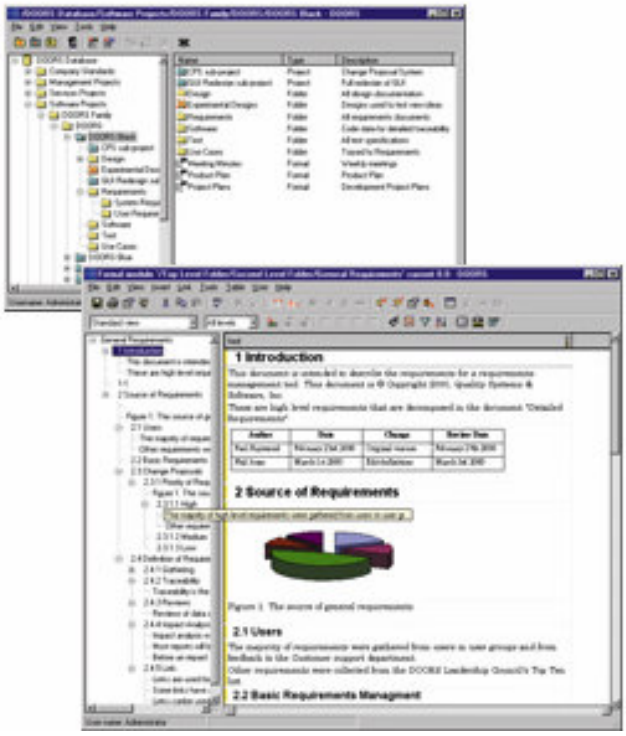


Figure 3C.12.1-3 Requirements Management Tool - *DOORS not only stores requirements, but also provides traceability to the designs.*

DOORS provides a single repository for the system, and eventually software, requirements. A common repository is important, as it brings together the requirements from a variety of sources.

DOORS also provides forwards and backwards traceability between requirements and:

- Test conditions
- Software designs, like a use case realization diagram in UML[6]

Traceability requires standards as well as tool support.

[6] UML is the Unified Modeling Language, and a Use Case Realization diagram is one of our work products in the Software Architecture. See section 3C.12.4 Software Architecture Design.

- Hardware designs, like a deployment diagram for servers in the production simulation environment
- Service Level Agreements (SLAs), which are arrangements that detail the quality of service metrics to be achieved

Traceability requires more than just tool support, however. It also requires standards for establishing the relationship between a requirement and its corresponding test conditions and designs. That is, how they link together. In conjunction with importing the requirements, the team creates these standards for CCSAS CSE.

3C.12.1.5 Proven Process

Our disciplined process to gather and analyze requirements follows a use-case driven approach, as defined by RUP. The alliance's process for System Requirements Analysis includes the following activities:

1. Identify System Requirements Decision Makers
2. Conduct Team Training
3. Categorize System Requirements

After this step, requirements will have been categorized as either for Version 1 or Version 2 analysis. For Version 1 and Version 2 separately, we then:

4. Analyze System Requirements
5. Prioritize System Requirements
6. Create the IEEE-Based System/Subsystem Specification
7. Informally Review the System/Subsystem Specification
8. Begin System Architecture Design
9. Acceptance of the System/Subsystem Specification

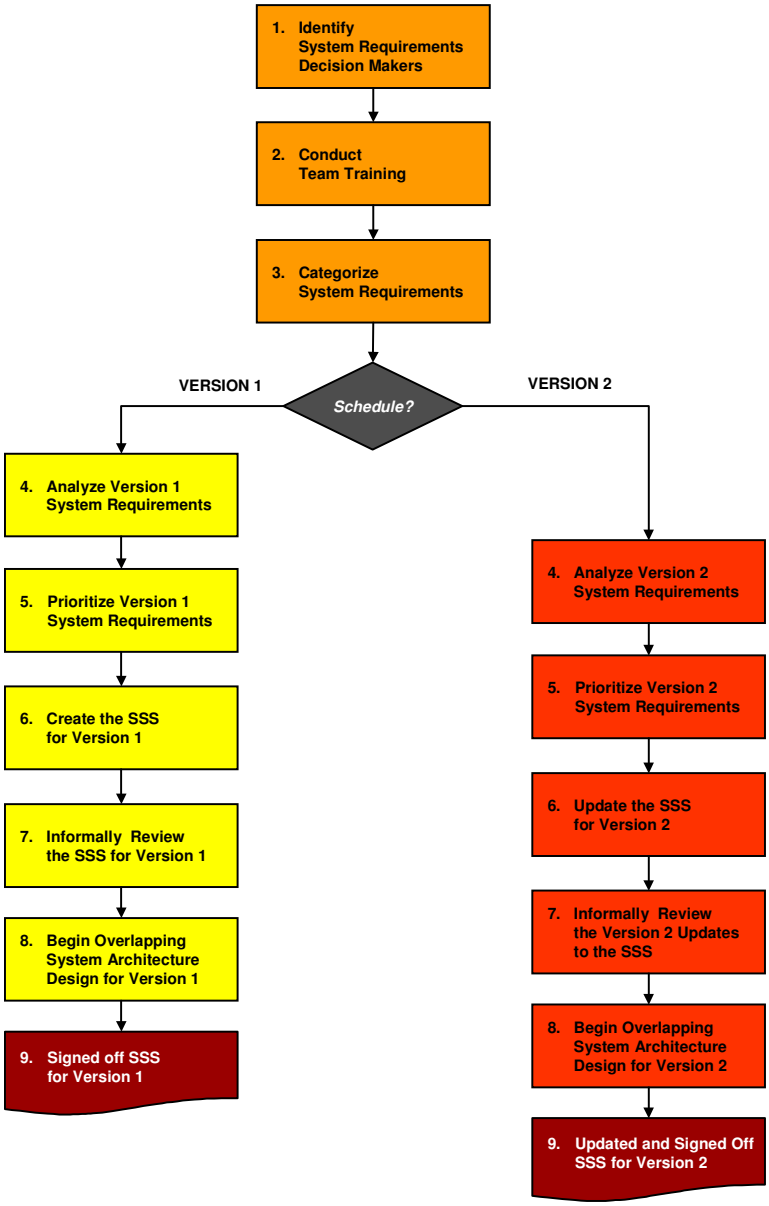


Figure 3C.12.1-4 System Requirements Analysis Process - *Analysis splits once requirements have been categorized for either Version 1 or Version 2. Version 2 analysis then proceeds in parallel, but on a different timeframe than, Version 1 analysis.*

We repeat steps 4 through 9 (above) for each version. For example, there are two separate System Requirements Reviews, for acceptance of Version 1 and Version 2 requirements.

The following paragraphs describe this process.

3C.12.1.5.1 Identify System Requirements Decision Makers

We work with CCSAS project staff to determine which CCSAS project, LCSA, and DCSS decision makers to involve in analysis. These groups empower their respective decision makers to speak for the project staff and users whom they represent. This includes the authority for decision makers to sign off requirements on their behalf.

As an example, a candidate list of groups to involve in the effort might include:

- End users
- Functional Subject Matter Experts
- Project sponsors
- Enterprise security architects
- External interface liaisons
- Enterprise data analysts and DBAs
- Application designers and builders
- User trainers
- Testers
- Operations support personnel

We work with CCSAS project staff to confirm these groups and identify specific individuals to collaborate with the specialized analysis team. We expect to have a sufficient number of decision makers, as follows:

- Decision makers for case management requirements
- Decision makers for financial management requirements
- Decision makers for technical requirements
- Decision makers for state-level functions (which includes decision making responsibility for integration with external agencies)
- Decision makers for SDU integration

CCSAS project, LCSA, and DCSS staff should also plan for analysis tasks.

Keeping the decision making team to a reasonable number provides adequate representation while remaining efficient.

The CCSAS project, LCSAs, and DCSS shall explicitly allocate enough time for each decision maker to meet with the requirements analysis team. This includes participation in weekly requirement review meetings and in analysis sessions. The latter could last anywhere from a half day to several weeks in duration.

3C.12.1.5.2 Conduct Team Training

The formal training for the specialized analysis team aims to:

- Improve the efficiency of the analysis process
- Improve the quality of and confidence in the requirements

The CCSAS project staff and alliance team members who analyze requirements need to understand the various analysis tools and practices. We conduct training sessions for them, on project-specific practices and procedures for:

- Writing requirements and requirements meeting facilitation
- UML use cases and use case development
- Rational Rose, the use case modeling tool
- Telelogic DOORS, the requirements management tool

We conduct one training session, lasting approximately one to two days, for each of these topics.

The alliance and CCSAS project, LCSA, and DCSS staff will schedule these one-time, highly specialized sessions. This training is specific to the Requirements phase of the base contract. The knowledge gained is not needed for maintenance of the system. Therefore, it is not a part of the formal Knowledge Management Plan. Additional training requests will be addressed using the Change Request Management process. Any reproducible materials used in this training will be given to the State.

3C.12.1.5.3 Categorize System Requirements

Categorizing system requirements includes the following steps:

1. Copy business requirements into draft system requirements
2. Perform an initial review of the system requirements
3. Categorize the system requirements

The following paragraphs describe these steps.

3C.12.1.5.3.1 Copy business requirements into draft system requirements

The business requirements are the key input to System Requirements Analysis. They represent the thinking of County and State subject matter experts and system users. Our analysis effort will confirm and refine those requirements.

At contract award, the State will have baselined the business requirements, stored in DOORS. The analysis team copies the business requirements into the draft set of system requirements. The copying step includes any basic formatting and transformations that are necessary to transform the business requirements into system requirements.. This draft set of system requirements then serves as the starting point for System Requirements Analysis. In this way, subsequent requirement clarifications affect the text of the draft system requirements, rather than the baselined business requirements.

We use the standards and procedures in Section 3C.5 Requirements Management.

3C.12.1.5.3.2 Perform an initial review of the system requirements

The analysis team then reviews the initial system requirements. This review gives the team the opportunity to clarify requirements and raise issues. The team works with the decision makers concerning redundant, confusing, or conflicting requirements. As such, new business requirements or changes to business requirements might arise. We address any such changes through our formal change management process, described in Section 3A.4 Change Request Management.

This review is not a formal review, as described in Section 3C.8 Technical Reviews.

3C.12.1.5.3.3 Categorize the system requirements

The analysis team categorizes each system requirement as:

- Version 1, Version 2, or Common, for scheduling purposes
- Summary or detailed, for organizing purposes
- Functional or technical, for classifying purposes

This categorization occurs in parallel with the initial review. That is, as the analysis team reviews requirements, they categorize them.

At this point, requirements will have been categorized as either for Version 1 or Version 2 analysis. Next, the analysis team executes the following steps (4 through 9) separately for Version 1 and Version 2. These steps occur in parallel, with Version 1 requirements analysis ending prior to Version 2 requirements analysis.

3C.12.1.5.4 Analyze Requirements

We now analyze requirements. (At this point in the process, we begin executing these steps separately: first for Version 1 and later, in a separate step, for Version 2. Recall the discussion above in the “Proven Process” overview.)

We analyze functional requirements and technical requirements differently. The following paragraphs explain the difference.

3C.12.1.5.4.1 Analyze functional requirements and create use cases

Having reviewed and categorized the functional requirements, we now more formally analyze them. The functional requirements have been enumerated in the requirements tool, essentially in a list. We have a hierarchical relationship – a vertical link – between summary and detailed requirements. But we lack a horizontal link that relates summary requirements with each other or detailed requirements with each other. From this perspective, the requirements are loosely affiliated, making them very cumbersome to analyze. Moreover, it is nearly impossible to discern the order in which business events occur to achieve system goals.

To solve this, the analysis team models the summary functional requirements with high-level use case diagrams. Use cases explicitly draw attention to relationships and provide order. They capture how requirements translate into user interactions with CCSAS CSE. At a high level, the team orders these interactions and their system responses.

The high-level CSE use case model represents system level functionality.

High-level use cases summarize events at a level consistent with the summary functional requirements. The high-level use cases provide the necessary functional context for System Architecture Design.

The use cases created in this step differ from the detailed use cases of Software Requirements Analysis. Those use cases describe discrete interactions between an actor and a software item of CSE. They describe the detailed events an actor participates in to achieve a business goal. Detailed use cases are driven from detailed functional requirements. As discussed above in the subsection on categorizing requirements, we wait to analyze detailed functional requirements until Software Requirements Analysis.

3C.12.1.5.4.2 Analyze technical requirements at a detailed level

Technical requirements cannot be translated into use cases, because use cases represent functionality and business flow. To analyze these requirements, the analysis team instead confirms that they have been specified in sufficient detail. Where necessary, the team creates detailed technical requirements to further explain summary technical requirements.

We need this detail for the System Architecture Design. Knowing that CSE must "respond in a reasonable amount of time" or "scale up to a large number of users" does not provide enough information for high-level design of hardware and software. One cannot size and procure a server machine with specifications like "we would like to buy a machine that must handle a large number of users." Instead, we need to know more details, like the number of users or response times for different types of transactions.[7] Because these requirements have widespread impact on the system design, they need to be expressed in objective terms.

3C.12.1.5.5 Prioritize System Requirements

Next, the analysis team works with decision makers to prioritize the requirements, thereby giving requirements context.[8] Priorities provide input to a variety of tasks, like identifying opportunities for prototypes or assigning the build-out of requirements to work plan increments.

[7] This is a discussion about how we categorize requirements. Ultimately, requirements need to be at an appropriate level of detail for subsequent design. This is what we refer to as "detailed requirements."
[8] Again, steps 4 through 9 of the analysis process occur separately for Version 1 and for Version 2. We do not prioritize across Versions.

Individual functional requirements, in their raw form, do not lend themselves to prioritization. They describe a single step to reaching a business goal. In contrast, use cases organize functional requirements by expressing relationships and order. Use cases are easier to prioritize as they describe a whole business goal. So the team assigns priorities to use cases.

We treat technical requirements differently from functional requirements for prioritization. This is because a technical requirement does not describe a step in reaching a business goal. Rather, this type of requirement describes qualities of service for any number of business goals.

We prioritize (functional) use cases and detailed technical requirements against a number of criteria, including:

- Business value
- Complexity
- Size
- Impact
- Level of understanding
- Propensity for change

Using various criteria makes the prioritization more useful. For example, we might design a use case with high complexity and low level of understanding earlier than other use cases. This helps identify issues, uncover dependencies, and gain a better understanding of associated work plan tasks. As another example, we might schedule a use case with a high likelihood for change later in the cycle to let the use case stabilize.

We document these priorities either in DOORS or in the work plan.[9]

3C.12.1.5.6 Create the IEEE-Based System/Subsystem Specification (SSS)

By now, we have created the constituent work products that comprise the SSS. This process step is an activity in versioning the baseline of those work products and packaging them into the SSS.

The SSS is an electronic cross-reference to the requirements and use case diagrams. These work products, and their traceability links, exist in the requirements and UML repositories. We version the work products under our configuration management processes. For a particular version of the SSS (e.g. the baseline), the cross-reference therefore needs to specify the appropriate version of the work product. As well, the

[9] Priorities help drive work planning efforts. As the alliance creates detailed procedures for work planning, it will determine whether storing these priorities in DOORS or in the work plan makes the most sense.

alliance provides procedures for staff from the CCSAS project and IV&V to locate these work products.

3C.12.1.5.7 Informally Review System/Subsystem Specification (SSS)

Decision makers review the SSS on behalf of the project staff and users they represent. This informal review does not aim to eliminate remaining issues or questions. Instead, this review confirms that the analysis team understands the highest priority requirements and use cases. Then, the development team can begin System Architecture Design to validate the requirements.

3C.12.1.5.8 Begin System Architecture Design

We begin System Architecture Design before official sign-off of the SSS. Overlapping this design with System Requirements Analysis is consistent with IEEE. This allows us to validate that the system requirements are consistent and feasible. We consider initial System Architecture Design to be part of System Requirements Analysis because feedback from the design affects the requirements.

3C.12.1.5.9 Acceptance of the System/Subsystem Specification (SSS)

The goal of System Requirements Analysis is the delivery of an SSS that contains requirements that are:

- Traceable to their sources (Business Requirements)
- Consistent with their sources (Business Requirements)
- Testable (with regards to technical requirements)
- Feasible of system architecture design
- Feasible of operation and maintenance

We submit the SSS for formal deliverable walkthrough. After we have identified and addressed major issues from this walkthrough, then:

- The SSS enters the formal CDL acceptance process.
- Upon acceptance, the SSS is baselined into the configuration management (CM) process. This does not preclude future changes to the SSS. We believe that requirements evolve throughout the project's lifetime, such as from legislative changes. We apply formal CM procedures to evaluate proposed changes for their impact on the project.

3C.12.1.6 Conclusion

The analysis model defines *what* a system will do or provide. It does not define *how* the system will be constructed – we define that later in the design model. The analysis model includes several deliverables and work products, from detailed requirements to use cases, leading up to the System/Subsystem Specification document. These jointly move the definition of the application to a greater breadth of understanding. Each work product offers a slightly different perspective or view of the solution. Taken together, they answer the "what" questions about the system.

The alliance performs a thorough analysis that:

- Uses a specialized team trained in requirements analysis
- Stores requirements in a tool that supports traceability
- Categorizes requirements for schedule and organization purposes
- Analyzes functional requirements with the Rational Unified Process and a use case-driven approach
- Brings to bear deep understanding of requirements from the successful implementation of Child Support systems in other states
- Validates requirements by overlapping analysis with System Architecture Design

Analyzing requirements is more than just writing down what the user says. We use a robust approach to derive a consistent set of well-defined requirements.

The System Architecture Design is a view of a system's features, high-level component model, and interactions with other systems. We use proven architecture design principles for hardware and software. We also leverage "best of breed" system design assets from other engagements. Creating the design uncovers inconsistent or missing requirements and determines the feasibility of others. As such, we create this design while continuing to analyze requirements. This overlapping approach results in higher quality and better-understood requirements – sooner rather than later.

3C.12.2 System Architecture Design

3C.12.2.1 Introduction

CCSAS CSE will be a large and complex system, containing thousands of moving parts. This magnitude and complexity requires a scalable computing infrastructure – like the one the alliance will deliver. Such infrastructures are engineered to handle complex transactions and high volume processing.

Application and hardware designers, however, are less effective when constantly immersed in the complexity of large systems development. So we break down systems into smaller, more comprehensible building blocks – blocks with manageable scope, which we can understand and talk intelligently about. We can then more easily design, build, configure, and maintain these blocks, often independently of each other.

System Architecture translates requirements into a blueprint for the major technical and functional components.

The engineering practice of decomposing a system, based on functional boundaries, consistent organizing principles, technical requirements, and proven approaches, results in a well defined system architecture. System architecture integrates high-level blueprints for a software project. Each blueprint represents a different viewpoint. A viewpoint models the system from a particular individual's perspective.

System Architecture Design identifies the hardware items, network items, software items, and manual operations in accordance with the system requirements. The design also identifies interfaces with external systems that are required to fulfill the system's needs. We develop the design using a set of proven design principles, to successfully meet the functional and technical requirements.

In addition to hardware and network design, the System Architecture Design includes a high-level functional architecture, which identifies large-grained software items. This provides a starting place for the subsequent Software Requirements Analysis and Software Architecture Design. These are discussed in Sections 3C.12.3 Software Requirements Analysis and 3C.12.4 Software Architecture Design, respectively.

Section 4A.3.2 CSE System Architecture represents the alliance's work to date on the System Architecture Design. This section, on the other hand, describes the formal process and proven principles we will use on future iterations of the design.

3C.12.2.2 Proven Principles

Any consistent, adaptable system architecture must be designed with a set of principles in mind. The alliance brings a great deal of experience designing and developing robust, flexible systems that implement tried and tested system architecture principles. This includes, but is not limited to:

Proven principles lead to proven architecture.

- Integrated View of Development, Execution, and Operations
- Architecture Layering
- Concentration on Technical Requirements
- Best of Breed Design Components
- High-level, Component-based Design
- e-Business Architecture Style
- Integrated Data Model
- Requirements validation using the System Architecture Design

The following paragraphs describe these principles.

3C.12.2.2.1 Integrated View of Development, Execution, and Operations

Section 4A.3.2 CSE System Architecture describes three different viewpoints – enterprise, informational, and computational – that together represent CCSAS CSE's functionality. The alliance believes as well in using multiple viewpoints, or blueprints, for representing the technical aspects required to deliver this system.

These system environments (which we also call architectures) are:

- **CSE Development Architecture,** which contains development tools, methods, standards, and procedures for building CCSAS CSE. It specifies a developer's toolbox, with the tools and techniques for building system software.

- **CSE Execution Architecture,** which contains the production hardware and software, run-time environments like the application server, and reusable technical services called by application code. Technical services might include database access, security, and logging, among others.

- **CSE Operations Architecture,** which provides system administration, monitoring, restart/recovery, backup, and other related production-time services.

We have a comprehensive view of architecture.

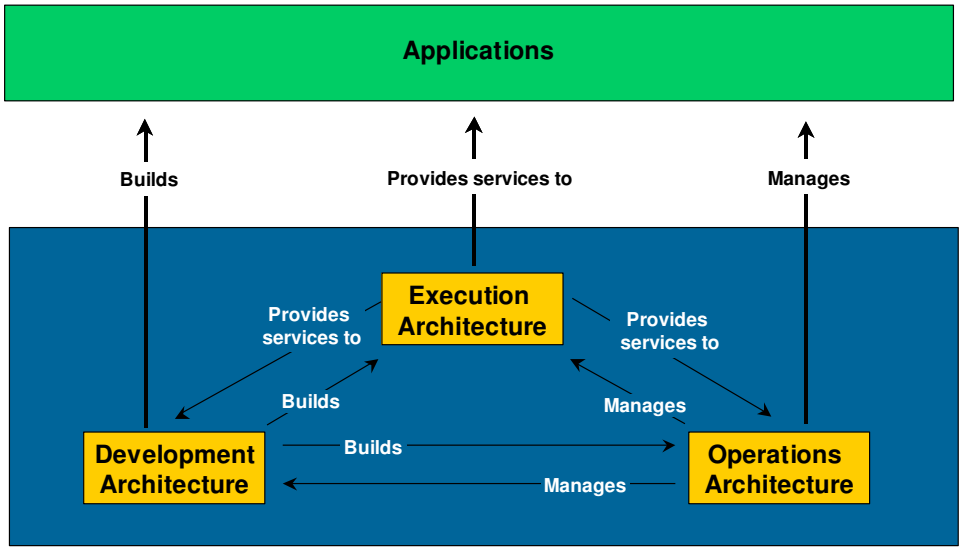


Figure 3C.12.2-1 Development, Execution, and Operation Architectures - *These architectures focus on services for coding, executing, and administering large systems.*

These three environments are highly integrated. We use the development architecture to build the execution and operations architectures. The operations architecture manages not only the execution architecture but also the development architecture. (We need to have a stable development environment with backup, monitoring, and other similar facilities.) The operations architecture relies on logging and other services that the execution architecture provides.

Section 3D Work Breakdown Structure refers to these three environments in the Technology Architecture tasks.

3C.12.2.2.2 Architecture Layering

The alliance uses fundamental layering principles to separate user interface logic, business logic, data access, messaging, networking, and hardware. This is consistent with the architecture principles outlined in Section 4A.3.3 CSE System Architecture Objectives.

Layering systems provides a number of advantages:

- Creates reusable logic
- Supports "fix in one place"
- Enables changes in one layer without impacting another
- Simplifies designing, building, and maintaining applications
- Avoids complex, spaghetti code
- Facilitates application plug-and-play
- Leverages existing alliance assets[10] and other proven approaches, patterns, standards, and technical services
- Improves scalability and performance
- Allows application team members to specialize

Layering organizes the large and complex space of systems building.

We divide CCSAS CSE into the following representative architecture layers (to be further defined and developed during the System Architecture Design activities):

- Presentation
- Business Logic
- Data Access
- Messaging
- Networking
- Hardware

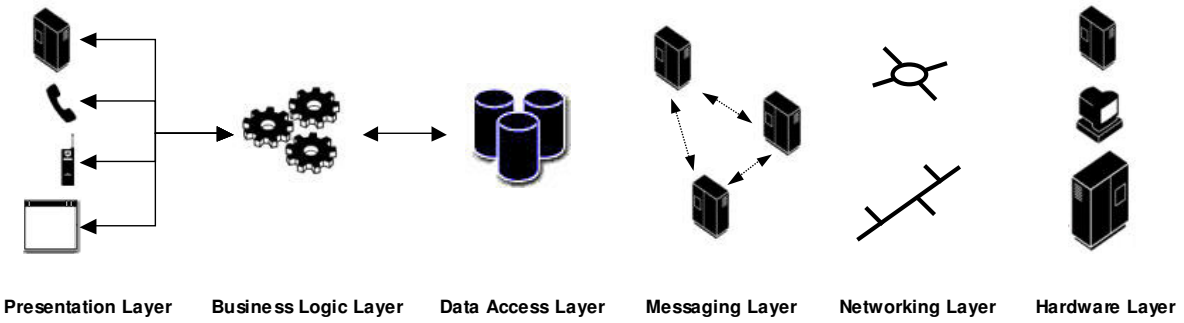


Figure 3C.12.2-2 Architecture Layers - *Architecture layering simplifies the fundamentally complex space of large systems building.*

The following paragraphs discuss these layers.

[10] Like GRNDS, our set of J2EE frameworks, which is discussed in Section 3C.12.4 Software Architecture Design.

3C.12.2.2.2.1 Presentation Layer

The presentation layer manages the human-computer interface. This includes capturing user actions and generating events, presenting data to the user, and managing dialog flow. Specifically, the presentation layer:

- Formats and displays data
- Captures user input
- Validates input format
- Responds to user actions
- Displays errors and warnings
- Manages content
- Provides user help

The presentation layer is thin, containing minimal business logic.

A presentation layer supports different types of user access, like HTML pages, WML pages for a web-enabled phone, Computer Telephony Integration voice prompts, and even batch scripts. It can be tailored by user type.

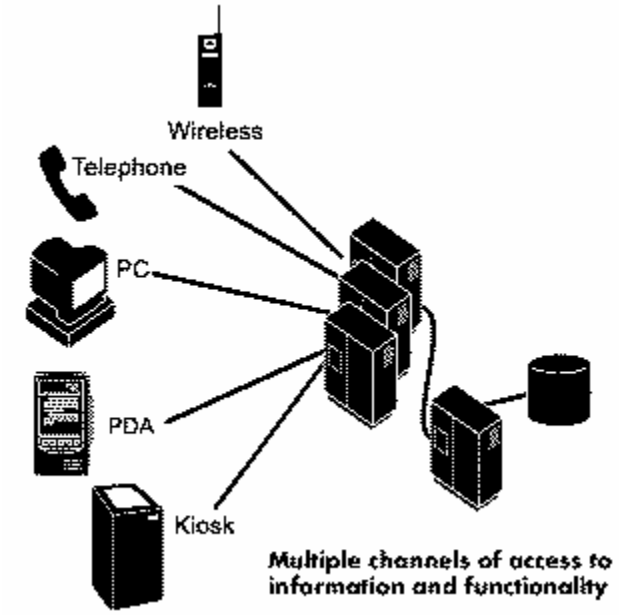


Figure 3C.12.2-3 Multiple Access Channels - *A well-layered architecture supports different types of user interfaces on the same, reusable business logic.*

This approach does not intend to deliver all of these different types of user interfaces for CCSAS CSE. The point is that we architect the system for extensibility. We build it to more easily support these user interfaces in the future.

3C.12.2.2.2.2 Business Logic Layer

The business logic layer expresses business rules, formulas, validations, and procedures. This might include the steps for making a case eligible for closure or the rules for allocating payments across an NCP's obligations.

Business logic is reusable across different presentations.

Business logic also includes the control structure that specifies the flow of processing for business events and user requests. The isolation of control logic improves a system's ability to adapt to changing business processing flows. The isolation of business rules improves a system's ability to adapt to changing business requirements, such as new legislation.

Business logic should be reusable across several different types of user interfaces. We explicitly recognize this in the software by separating it from the presentation layer.

Specifically, the business logic layer:

- Contains reusable business logic, across user and batch interfaces
- Validates input and raises errors and warnings
- Retrieves and updates data from application's data access layer
- Manages transactions within an application
- Audits and logs activities

3C.12.2.2.2.3 Data Access Layer

The data access layer is responsible for accessing the data store, typically a relational database. This includes functionality to create, retrieve, update, and delete data.

A data access layer separates SQL from business logic.

Business logic rules and business processes are independent from the location and mechanisms for data storage. As such, this layer is separate from the business logic layer. This layer also includes architecture services and standards that provide more efficient data access. For example, large collections of data will be returned in subsets to the user, for improved response time.

Specifically, the data access layer:

- Performs retrieval and updates to the data store
- Maps data from its tabular or other format into business object data structures
- Resolves locking issues when multiple users try to access the same data
- Supports referential integrity, whether at the application or database level

3C.12.2.2.2.4 Messaging Layer

The messaging layer enables the system to interact with other systems, giving access to externally stored data and externally executed functionality. It hides the transport protocols from the application, which simply declares what data or procedures it needs

from other systems. We use proven Enterprise Application Integration (eAI) approaches and tools for this layer. See Section 3C.12.8 System Integration.

Specifically, the messaging layer:

- Integrates systems, within and outside CCSAS CSE
- Routes and sends outgoing messages
- Routes and receives incoming messages
- Provides queued message delivery
- Formats and transforms messages
- Validates messages
- Subscribes to and publishes enterprise-wide events

The messaging layer does not contain business logic, except for message routing, formatting, transformation, and validation.

A key feature of the alliance solution is our well-architected messaging layer. We separate formatting, validation, transformation, and routing logic into their own eAI modules. This contrasts with a more traditional approach, in which that logic resides in the applications that send and receive messages. Our approach – using encapsulated eAI modules for that logic – allows us to change integration rules independently of the business applications. This provides maintainability and extensibility, directly addressing the CCSAS architecture attributes provided by the State. See Section 3C.12.8 System Integration.

3C.12.2.2.2.5 Networking Layer

The networking layer is the communications system that connects the components of the hardware and software layers in the system. Our experienced technical architects perform a thorough analysis of the required communication needs. They confirm the feasibility of technical network requirements and design.

3C.12.2.2.2.6 Hardware Layer

The hardware layer supports the deployed software in production. It considers the servers, clients, peripherals (like printers and scanners), storage, and networking hardware required to support the system. Consideration is given to system architecture principles as well as required capacity, scalability, price, vendor viability, and system availability.

3C.12.2.2.3 Concentration on Technical Requirements

While the System Architecture Design demonstrates the implementation of system requirements, special attention needs to be paid to *technical* requirements. As defined in Section 3C.12.1 System Requirements Analysis, these requirements do not fall into the category of business functionality. Rather, technical requirements include performance, reliability, usability, and serviceability. The system-wide decisions made during this phase are driven by – and significantly impact – these technical requirements.

Our use of technical pilots early on is consistent with the Rational Unified Process (RUP).[11]

[11] See Section 3C.11 System Life Cycle Model, which introduces RUP.

We carefully analyze these requirements to check their feasibility. This can be accomplished with a "pilot" to test the fundamental system architecture. Where necessary to prototype technical requirements, we build simple, incomplete applications that mimic the system architecture. We then conduct performance and reliability tests, modifying the system architecture as necessary to meet the specified requirements.

3C.12.2.2.4 Best of Breed Design Components

Our innovative response to California's CSE automation challenge supports the State's vision for a state-of-the-art solution. Reusing best of breed design components and reusable frameworks will reduce development risk while preserving the benefits of constructing the solution on a modern platform. We incorporate system-level designs from other proven engagements into the CSE System Architecture.

3C.12.2.2.5 High-level, Component-based Design

We use component technologies and design approaches for the CSE System Architecture. Component technologies are more applicable to software architecture, but we start applying component principles now, to lay the groundwork for software design.

A component-oriented design breaks down the system into functional components. Each component concentrates on a particular area of functionality. Components may interact with each other but only through a set of well-defined interfaces and only through standard communication protocols.
A component-oriented design provides for easier design and coding of the component. Component interfaces are determined at design time. The development team building the component focuses on satisfying the interface contract. The component's internal workings should have no affect on other components, as long as the interface contract is satisfied.

A component-oriented approach is fundamental to Architecture Design.

Additionally, a component-oriented design better supports maintenance. Changes can be isolated to a single component as long as the component's interface does not require modification. Even if it does, the explicit interfaces better support impact changes.

A reader at this point might think: "Sounds good. Build me some of those components. But let's get back to the fine art of system architecture."

But components *are* a part of system architecture. We know that to build reusable, maintainable, and high performance business components, we need to define them early.[12] In other words, our system architecture needs to reflect the fact that we are using components. For, components should not be an afterthought. They are fundamental to the way the alliance thinks about, communicates, evaluates, and

[12] Techniques for designing in reusability, maintainability, and performance are more clearly defined in Section 3C.12.4 Software Architecture Design and, more generally, Section 4A.3.3 CSE System Architecture Objectives.

ultimately designs CCSAS CSE. A component-based approach forms the foundation for our best-of-breed solution.

Because we are confident in this approach, we have introduced business components in the CSE System Architecture. And this is more than a mere introduction – we have based our entire architecture description around this approach.

The SCP says that the CSE System Architecture viewpoints "shall identify key items in each that are related to items in other viewpoints." Business components provide the unifying theme that the State seeks across the three different viewpoints:

- A component embodies business policies and rules. This is the **Enterprise Viewpoint**.
- That same component encapsulates data – in particular, data related to its business policies. This is the **Informational Viewpoint**.
- Finally, that same component has interfaces and interacts with other components. These interfaces allow others to retrieve the component's internal data and execute the component's business policies and rules. This is the **Computational Viewpoint**.

A component architecture supports the viewpoints in the CSE System Architecture.

In this manner, components tie together the viewpoints. A high-level component model is a key work product of our System Architecture Design. Section 3C.12.4 Software Architecture Design describes components in more detail

3C.12.2.2.6 e-Business Architecture Style

The e-Business architecture style prescribes a thin client approach. This uses a browser-based user interface, built with Internet standards like HTTP and HTML. Any user can access the system with a minimal amount of software: an operating system and a standard web browser like Microsoft's Internet Explorer. This can reduce software distribution costs, centralize software changes, and increase the ability to reach the system from multiple locations.

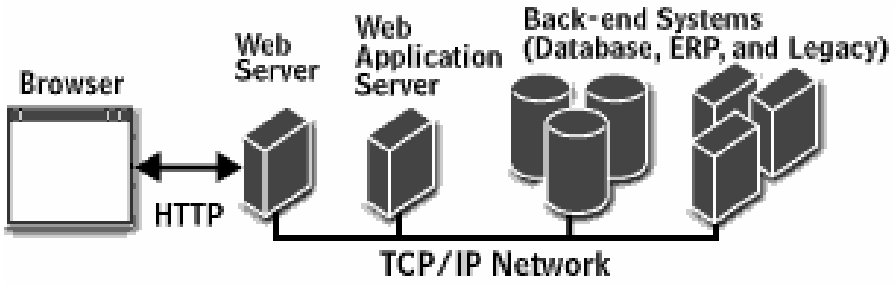


Figure 3C.12.2-4 e-Business Architecture Style - *A thin client uses well-recognized Internet-style user interfaces. It is accessible from multiple locations.*

The e-Business architecture style with application servers has the following advantages:

- **Scalability.** When additional hardware processing power is required to support a heavier load of transactions or users, an e-business architecture can scale vertically (by adding more memory or CPU's to existing servers) or horizontally (by adding more servers to the cluster to share the processing load). Two-tier architectures typically can only scale vertically, which eventually reaches the performance limits of a particular server. Scaling involves not only hardware, but also software. An application server allows multiple instances of application components to run at one time. As we add more hardware servers, we can run more copies of these components.

- **Load Balancing.** By using a cluster of physical servers at the application tier, intelligent distribution of user requests to the least busy server or servers allows the application as a whole to respond better under heavy load.

The e-Business architecture style supports CSE architecture best practices.

- **Fault Tolerance / Failover.** The clustering ability of e-business architectures allows one application server to take over for another in the event of a hardware failure. While this often results in reduced overall system performance during a component failure, the site as a whole remains up and functioning.

- **Maintainability.** By breaking the application up into distinct layers or tiers, changes to one portion of the application have a reduced impact to rest of the application. In addition, since the client tier is responsible only for presentation, changes to application logic are made once at the application server tier and are automatically used by all clients. This contrasts with two-tier architectures, which include business logic on the client tier where application changes would have to be made to all clients.

- **Support for multiple access channels.** By separating the presentation from the business logic and enterprise data, the same business function can be reused by different access channels. For example, both a browser and a PDA could make a request to the same business logic on the application server tier and yet display the results differently based on the platform display requirements. This is also a benefit of the architecture layering discussed above.

- **Application Performance.** Application servers typically include performance-enhancing features such as caching for object instances, database connections, and network connections. An application server intelligently pools scarce resources and matches them with client requests. These would either be unavailable or would need to be custom-coded in a two-tier architecture.

- **Hardware Performance.** Hardware can be purchased and tuned to fulfill a particular role such as transaction processing (which typically requires more memory or CPU's) or data management (which typically requires more I/O capability or physical data storage). The ability to tune different servers for different purposes improves overall system performance.

- **Network Performance.** The application and data tiers are often located together in a data center with high-speed network connections. This co-location improves the communication between these two tiers. Requests passed over the network from the client tier (which is typically a lower speed network connection) are often small requests for services instead of large requests for data, conserving network bandwidth.

Physical components play a critical role in e-Business architectures. Components are encapsulated units of executable software. As such, they can be distributed throughout a heterogeneous environment like the Internet.

3C.12.2.2.7 Integrated Data Model

CCSAS CSE data can come from multiple sources, including the system's own RDBMS, the SDU, government agencies, and employers, to name a few. As such, CCSAS CSE must handle disparate data models. For example, the data representation of an interstate referral from CSENet may differ from the CCSAS desired representation.

To handle this situation, the alliance proposes developing an integrated data model and data dictionary. We suggest using industry standards like Common Warehouse MetaModel (CWM) to create this model. See Section 3C.12.6 Database Design and Development.

An enterprise-wide data dictionary provides a consistent, reusable data model – regardless of data source

The data access and messaging layers translate data from external systems and databases into the format expected by CCSAS CSE. This allows for much simpler business logic code. In addition, the decoupling of the enterprise-wide data model from the data source reduces the dependencies on any particular external system or database.

3C.12.2.2.8 Requirements Validation Using the System Architecture Design

In a strict application of waterfall methodology, projects analyze and sign off every requirement before beginning design. But experience has shown that it is very difficult to confirm every gathered requirement before having a better feel for the system design. In many cases where requirements precede design, changes to the requirements surface during the design phase.

We propose a better approach. We use the results of the system design to validate the requirements, following an overlapping process. Shortly after beginning requirements analysis, we begin work on an initial system design. We review the initial system design against the system requirements to determine if the requirements are:

- Correct
- Inclusive
- Feasible

We refine the design and refine the technical requirements for meeting the State's fixed business requirements. A stabilized design accounts for the identified requirements and appears feasible. In other words, we can build the design.

System design verifies system requirements.

This approach allows for clarifications to requirements that come with design, while letting the design and construction work progress. It also offers the system design as yet another tool to analyze and sanity check the requirements.

The standard "V-Model" of application development illustrates this approach. It places feedback loops after the phases on the left side of the "V."

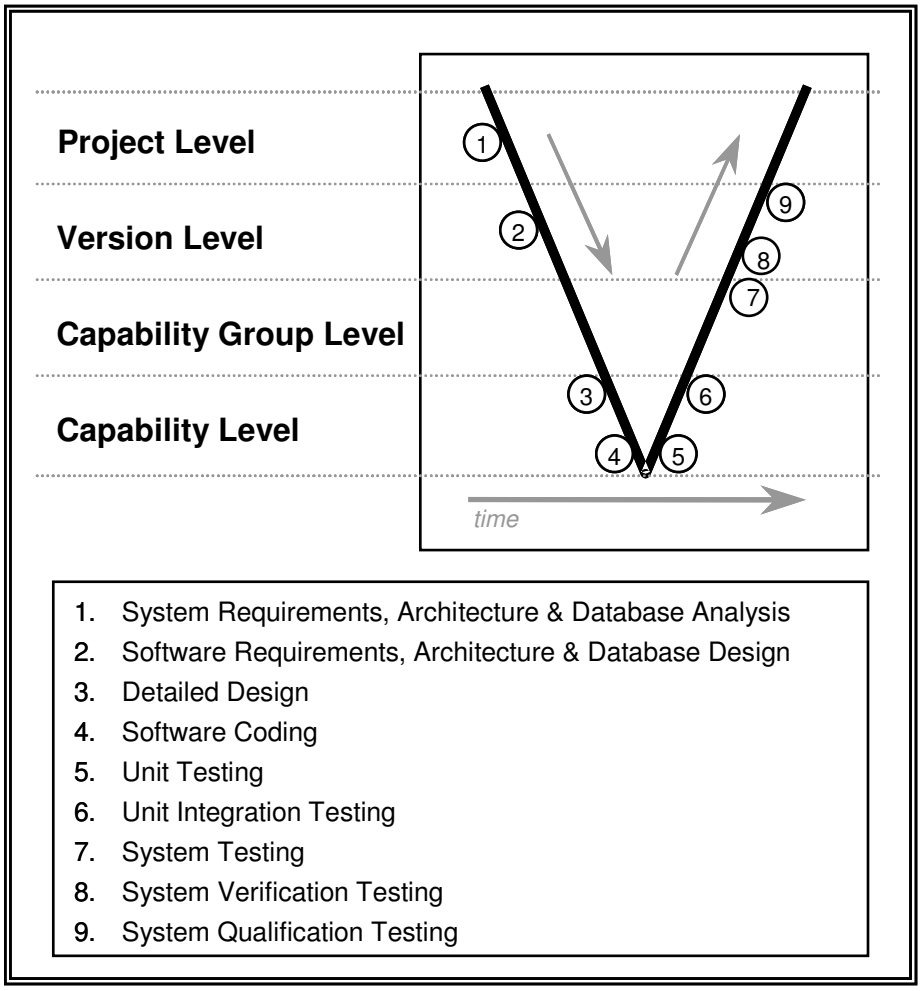


Figure 3C.12.2-5 V-Model - *We use the system architecture to provide feedback on unclear or inconsistent requirements.*

Our testing approach, discussed in Section 3C.12.9 Test Management, also uses the V-model.

3C.12.2.3 Proven Work Products

Work products of System Architecture Design include:

1. IEEE-based System/Subsystem Design Description (SSDD)

2. IEEE-based Interface Design Descriptions (IDDs)
3. Technical Infrastructure Description
4. System Performance Budget

The following paragraphs discuss these work products.

3C.12.2.3.1 IEEE-Based System/Subsystem Design Description (SSDD)

The IEEE-based System/Subsystem Design Description (J-STD-016-1995, Annex G.2.1) captures system architecture views. The SSDD describes the hardware items, network items, software items, manual operations, and interfaces that we build to fulfill the system requirements. The SSDD serves as a communications medium between the System Architecture Design team and the other project participants.

While conceptually a single document, the SSDD is delivered as a collection of references to specific design details. This complies with the IEEE standard, which does not predetermine the actual format or delivery mechanism for the SSDD. The SSDD cross-references to multiple design documents, stored in their native design repositories.

The SSDD links to multiple design documents, stored in their native tool.

We use configuration management processes and standards for the SSDD. Both the SSDD and its constituent work products receive unique identifiers for versioning. See Section 3C.4 Configuration Management.

The following paragraphs describe the sections we deliver with the SSDD.

3C.12.2.3.1.1 System-Wide Design Decisions

This section describes key features of the system's behavioral design. This includes how the system performs from a user's point of view, ignoring the internal implementation of software items.

3C.12.2.3.1.2 System Architectural Design

This section identifies and describes:

- Relationships between system components (hardware items, software items, and manual operations)
- How the components dynamically interact
- Interface designs

A key part of this design is the high-level business component model. As mentioned, Section 4A.3.2 CSE System Architecture contains our initial thinking on this model.

3C.12.2.3.1.3 Requirements Traceability

This section contains traceability from each system component to its preceding system requirements. As mentioned in Section 3C.12.1 System Requirements Analysis, we use a requirements management tool that provides traceability from requirements to portions of the design model.

3C.12.2.3.2 IEEE-based Interface Design Descriptions (IDDs)

The IEEE-based Interface Design Description (J-STD-016-1995, Annex G.2.2) captures the contract of exchange between CCSAS CSE components and external systems, like the SDU.

We identify each such interface. We specify how the interchange occurs, including information on sender, receiver, message formats, frequency, and parameter types and lengths. Ultimately, the design of these interfaces uses the enterprise-wide data dictionary, mentioned earlier.

The format of these interfaces is driven by our use of eAI (Enterprise Application Integration) technologies and approaches. Section 3C.12.8 System Integration discusses eAI and the use of XML for these formats.

When associated system requirements exist, we provide traceability from the interface to those requirements. It is possible for interfaces to exist solely for design purposes – such as decomposition. In these cases, no system requirements map to the interface. For rationale, see IEEE J-STD-016-1995, Annex G.2.2, Clause 4a.

The CSE/SDU Interface Design Description will contain the allocation of timeframe budgets between the CSE and SDU applications necessary to meet the two-day payment disbursement timeframe.

3C.12.2.3.3 Technical Infrastructure Description

The Technical Infrastructure Description (CDL TM018) defines the strategy, implementation and high-level plan for the CSE technical infrastructure. Technical Infrastructure incorporates the hardware and system software that constitutes a system environment. It does not include the developed application code. The TID is developed as a system-level product, but is completed in parallel with software design and detailed design activities. The Technical Infrastructure Description includes a description of the:

- Fundamentals behind the use and implementation of the technical infrastructure;
- Physical infrastructure (servers, LPARs, firewalls, routers, circuits, storage, etc.);
- Infrastructure configuration (how the system software is configured);
- Application server design;
- Database design (infrastructural); and
- A high level plan for the initial infrastructure implementation.

3C.12.2.3.4 System Performance Budget

The System Performance Budget relates performance requirements to appropriate subsystems, or architecture layers.[13] Consider a requirement that online transactions of a particular type have an end-to-end response time of a certain number of seconds. A budget might distribute a portion of that time to each of:

1. Network communication from the browser to the web server
2. Processing time for the Java Server Page in the web server
3. Network communication from the web server to the application server
4. Processing time for the Java components in the application server
5. Network communication from the application server to the database server
6. Processing time for the database to execute the SQL

Allocating performance requirements in this fashion helps us design our hardware, network, system software, and overall application model. For example, network architects need to validate that their proposed network link from the web server to the application server will be fast enough and have adequate capacity to stay within its budget.

This budget is useful not only for design, but also for testing. It helps us confirm that certain technical components meet their portion of the performance requirements. Moreover, when testing shows that the budget for a particular requirement has been exceeded, we can quickly focus in on the “offending” software modules or hardware that exceed their part of that budget. It is in those areas that we believe we have room – that is, budget – for improvement. The approach to developing the System Performance Budget is defined in the System Engineering Management Plan (CDL TM 001) and the final budgets are documented in the Performance Capacity and Management Plan (CDL TM 086).

3C.12.2.4 Proven Process

With our two-phased approach, we perform System Architecture Design twice. We use the System/Subsystem Specification (SSS) of Version 1 for the System Architecture Design of Version 1. Similarly, we use the *updated* SSS of Version 2 for the System Architecture Design of Version 2. See Section 3C.12.1 System Requirements Analysis. For illustration, we deliver the System/Subsystem Design Description (SSDD) twice. We deliver the first version at the end of Version 1 System Architecture Design and then an updated version at the end of Version 2 System Architecture Design.

Similarly, the downstream development stages – from Software Requirements Analysis through Testing – proceed separately for Version 1 and Version 2.

Version 2 builds upon Version 1, which we engineer to be flexible and extensible for Version 2. In this manner, we avoid rework. Moreover, overlap is minimal for both

[13] Here, we use “layers” as defined earlier in this section.

versions of System Architecture Design. This eases configuration management: we avoid having to manage concurrent efforts on the same work products across both versions.[14]

We follow the same straightforward, controlled process for Version 1 and Version 2 System Architecture Design:

1. Identify System Architecture Design Decision Makers (Version 2 may introduce new decision makers)
2. Adopt System Architecture Design Principles (Version 2 may introduce new principles)
3. Create Work Products
   3.1. Create IEEE-Based System/Subsystem Design Description
   3.2. Create IEEE-Based Interface Design Descriptions
   3.3. Create Technical Infrastructure Description
4. Informally Review Architecture Design Along with Requirements
5. Obtain Initial Concurrence for Design

[14] Managing concurrent efforts requires "branching" -- simultaneously maintaining two different versions of the same configurable item. Branching is a challenge, because the changes in one version later need to be merged into the other.

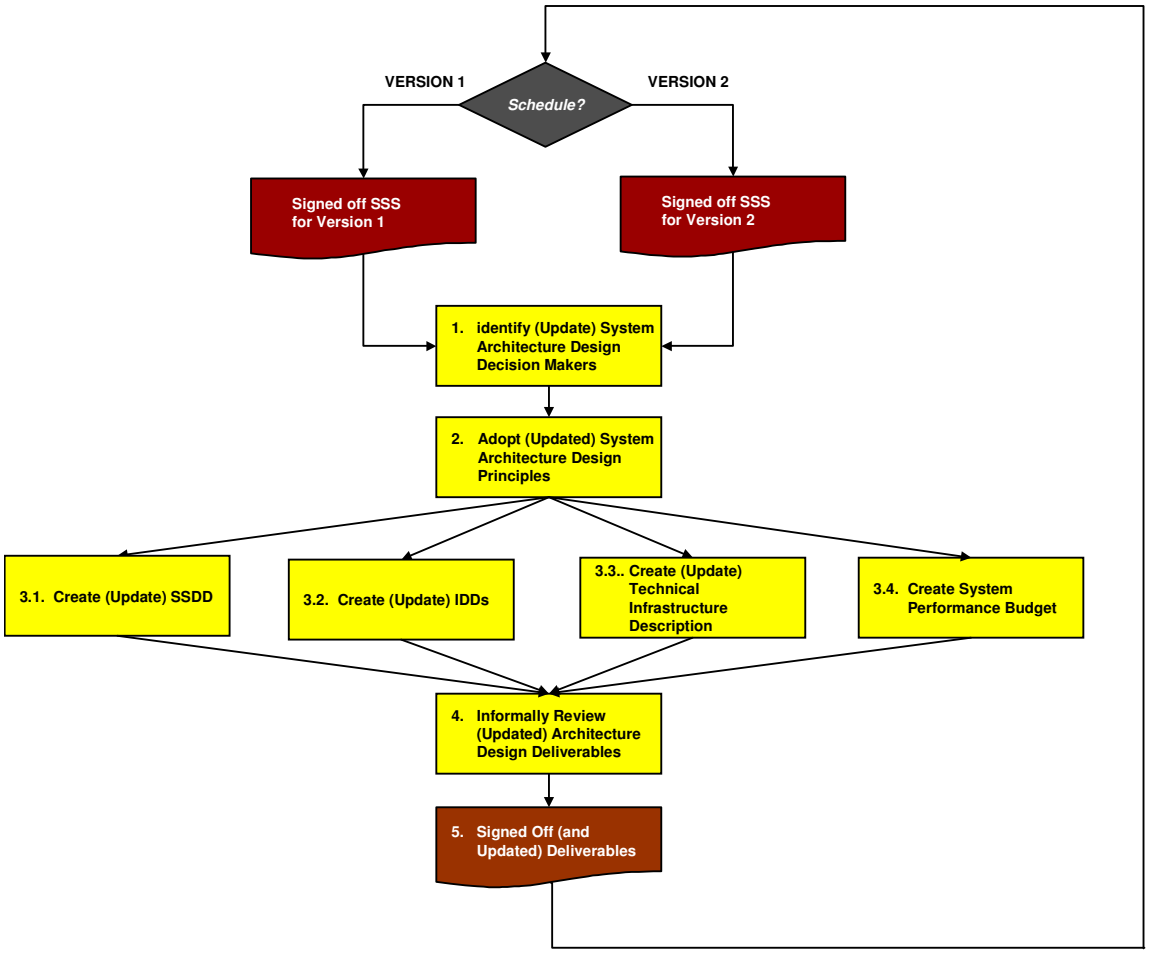


Figure 3C.12.2-6 System Architecture Design Process - *We create the SSDD for Version 1 and then repeat the tasks for Version 2.*

The following paragraphs describe this process.

3C.12.2.4.1 Identify System Architecture Design Decision Makers

The System Architecture Design decision makers are the individuals who review and concur with the submitted system architecture design.[15] The decision makers are most likely the same individuals participating in the System Requirements Analysis phase. We work with the CCSAS project, State, and DCSS to identify their respective decision makers before beginning the design process.

Decision makers who were not involved in analysis should become familiar with the requirements before the design process begins.

[15] The authority of these "decision makers" does not replace the formal Contract Deliverable Acceptance Process (CDAP). Rather, the decision makers provide concurrence on the State's behalf, before the deliverables enter CDAP.

3C.12.2.4.2 Adopt System Architecture Design Principles

To develop a consistent design, it is important to identify a set of system architecture design principles. Many such principles are discussed elsewhere in this document. These principles should be reviewed and agreed upon by the decision makers.

This can be tricky, as architecture principles are often a source of debate. This step is not meant to engage decision makers and development team members in philosophical discussions of broad, high-level principles or their relative importance. Instead, the principles should be documented in specific terms, relative to CCSAS CSE. Focusing on system specifics frees the decision makers to agree on a set of valuable principles.

Architecture principles set the stage for a smooth design process.

The initial set of principles comes from:

- Principles mentioned in this section
- The CSE Architecture Principles
- SCP Exhibit 3K
-

These principles influence system architecture and software architecture design. The System Engineering Management Plan describes how the CCSAS project architects will capture the principles, roll them out to system and software designers, and enforce them in informal reviews.

3C.12.2.4.3 Create Work Products

We create the work products as defined in the “Proven Work Products” subsection. The following paragraphs describe these steps, which occur in parallel.

3C.12.2.4.3.1 IEEE-Based System/Subsystem Design Description (SSDD)

The IEEE-based System/Subsystem Design Description is the first document to be created. Each requirement should either map to a particular system component or should be included in the system-wide design decisions section of the document. Traceability between the SSDD and the system requirements should be clear.

3C.12.2.4.3.2 Create IEEE-Based Interface Design Descriptions (IDDs)

Concurrently with the SSDD, we create IEEE-based Interface Design Descriptions. The SSDD refers to these IDDs. Each IDD captures a single interface between the CCSAS CSE application and external systems, like the SDU. The IDDs are shaped by joint activities, like interface design sessions and reviews, that the alliance conducts with external agencies. Section 3C.12.8 System Integration discusses these joint activities, with a special focus on the SDU vendor.

Some agencies and systems, like for CSENet, FCR, and FMS, have already defined fixed formats. In those cases, rather than creating an IDD to duplicate already existing information, we refer to the agency's documentation for the interface or to external agency interface agreements. In the event that documentation is incomplete, we create an addendum for our System Architecture Design.

3C.12.2.4.3.3 Create Technical Infrastructure Description

The architecture team creates the Technical Infrastructure Description. This can occur concurrently with the creation of the SSDD, IDDs, and the SDD.

3C.12.2.4.3.4 Create System Performance Budget

By now, we have analyzed the technical requirements in 3C.12.1 System Architecture Design. This budget spans software, hardware, and network items. As such, it requires input across the various architecture layers, from web, component, integration, hardware, and network architects.

This budgeting exercise has limited resources—for example, a limited amount of seconds for end-to-end response time, or a limited amount of space in the database. The architects from the different backgrounds are expected to lobby for sufficient budget for their own areas. For effective and reasonable budgeting, the alliance bases the allocation on facts: previous experience with these technologies on large projects like CCSAS. Moreover, our Chief of Architecture moderates budget debates with skillful negotiation and consensus building.

This budgeting exercise occurs concurrently with the creation of the SSDD, IDDs, SDD, and performance testing. For, technical architects need to know the requirement budgets for their particular components to design them.

3C.12.2.4.4 Informally Review Architecture Design Along with Requirements

We submit the drafts of the SSDD, IDDs, and Technical Infrastructure Description for review by the decision makers. As with the requirements review process, this initial review does not mean to obtain sign-off. Instead, its goal is raising questions and feasibility issues. We document issues at the initial session. But we resolve them in smaller break-out sessions with the appropriate parties.

Much as the design uncovers questions and issues with the requirements, the actual construction of the system will uncover questions and issues with the design. To address these questions as soon as possible, we complete the design only to a point where construction can begin. The earlier detailed design and construction begin, the sooner we validate the design and requirements in actual software and hardware.

> The purpose of the CCSAS project is not to produce a design, but rather to produce a system.

3C.12.2.4.5 Acceptance of Design

The goal of System Architecture Design are to have a product that is:

- Traceable to the system requirements
- Consistent with the system requirements
- Appropriate considering the accepted design principles and standards
- Feasible of the system components fulfilling their allocated requirements
- Feasible of operations and maintenance

The decision makers use these criteria to participate in the System Design Review (SDR).[16] We then submit the System Architecture Design for formal deliverable walkthrough. After we have identified and addressed major issues from this walkthrough, then:

- The System Architecture Design enters the formal CDL acceptance process.
- The accepted design is baselined into the configuration management (CM) process. This does not preclude future changes to the design. Proposed changes to both system requirements and architecture design will be channeled through the process described in Section 3C.4 Configuration Management.

Accepted design goes to configuration management, anticipating changes throughout development.

3C.12.2.5 Conclusion

Overlapping analysis and design results in a robust architecture that meets clearly analyzed requirements.

The alliance uses proven architecture principles to create the System Architecture Design. These include layering, reusable assets, component-based design, enterprise-wide data management, and an e-Business architecture style. We also build upon the architecture best practices clearly articulated in the SCP.
We overlap System Requirements Analysis and System Architecture Design. This confirms and validates requirements. As well, we use pilots where required to check feasibility; that is, whether or not we can actually build to meet particular requirements. This process improves the quality of – and confidence in – system requirements. When System Architecture Design occurs in conjunction with Requirements Analysis, we engineer the system solution for success.

[16] See Section 3C.8 Technical Reviews.

3C.12.3 Software Requirements Analysis

A software analysis model defines <u>what</u> software does. It does not define <u>how</u> the software is constructed. We define the "how" later in the design model. Our robust software analysis model includes several component-based work products, from detailed requirements to use cases to the IEEE-based Software Requirements Specification document. We also introduce low-fidelity prototyping, as a proven way to determine user interface requirements. Each work product offers a slightly different perspective of the solution. Taken together, they answer the "what" questions about the software.

3C.12.3.1 Introduction

The initial System Architecture Design (described in Section 3C.12.2) defines the hardware items, software items, and manual processes to be delivered in CCSAS CSE. The software items have been documented merely as large-grained components that provide high-level visibility to their functionality and structure. As with the *system* definition, *software* definition calls for a requirements analysis and architecture design for each software item identified.

Technical requirements have a larger influence on the *system* design while functional requirements have a larger influence on the *software* design. As such, during System Requirements Analysis and System Architecture Design, we focused more on the technical requirements than the functional requirements. We emphasized the detailed performance, availability, maintenance, and usability requirements and how they would affect system design. These requirements, and the resultant system architecture, carry over to this stage of development. They influence our software analysis, as our understanding of what the software should do is both enabled and constrained by the system architecture.

During Software Requirements Analysis, attention now shifts to CCSAS CSE's detailed functional requirements. What the software does and how it interacts with users becomes the focus. As mentioned, technical requirements remain part of the software equation. We also revisit them in Software Architecture Design, to validate that the software can be implemented on the system architecture. However, for the purposes of Software Requirements Analysis, we focus on functional requirements.

Software analysis focuses on functional requirements.

Regardless of whether dealing with functional or technical requirements, a strong *analysis* is still the key. Anyone can *gather* requirements. The emphasis must be

placed on the analysis effort to better understand and transform those initial requirements into work products that can guide subsequent software design.

3C.12.3.2 Proven Principles

Whether we analyze software or system requirements, many of the same principles apply. As with System Requirements Analysis, we **continue to:**

- Use a specialized team trained in requirements analysis
- Store requirements in a tool that supports traceability
- Analyze functional requirements with the Rational Unified Process and a use-case driven approach
- Validate requirements using the Architecture Design (only now we use the software architecture as opposed to the system architecture).
-

Section 3C.12.1 System Requirements Analysis has already explained these principles. In addition, for Software Requirement Analysis, we introduce the following new principles:

- Drive Software Requirements from System Requirements
- Follow the User's Mental Model

The following paragraphs describe these two principles.

3C.12.3.2.1 Drive Software Requirements from System Requirements

The software items are based on system requirements. System requirements are at a level appropriate to define large-grained software items. However, they are not adequate to design the software architecture. So we refine these system requirements into software requirements.

Defining software requirements is not a search for new requirements. Rather, it is an effort to explain further the system requirements, by driving the functional and technical system requirements to a more detailed level. We perform an analysis on these requirements that translates them to the users' conceptual view of the user interface. We also link together related requirements to identify business processes. Finally, we prioritize these business processes to help with work planning (i.e., assigning the tasks to various work plan increments).

3C.12.3.2.2 Follow the User's Mental Model

We introduce low-fidelity user interfaces, as discussed in the "Proven Work Products" section below. While user interfaces are part of design, introducing them early drives out functional and behavioral requirements. Similarly, near the end of software analysis and during software design, we begin testing usability. Traditional usability testing relies on the following approaches:

- Usability tests include business representatives who are long-standing members of the development team. These business representatives provide input to the early interface designs. They are intimately familiar with the requirements and how the software builds out the requirements. The representatives then test out those same designs.

- During testing, a designer first explains how the user interface works. The designer then gives a business representative a task to complete. When the representative cannot easily complete a task, the designer points him in the right direction.

This traditional approach is straightforward and frequently used. And this is precisely why many systems are not usable, despite well-intentioned efforts. Granted, the aforementioned techniques can validate that the software builds out the functional requirements. But they do not really test usability, nor do they help predict user acceptance. This is because:

- In this context, business representatives have worked on the project for a while. They have immersed themselves in the daily project details. They have become accustomed to how designers think the user interface should look. The representatives have themselves contributed to the designs. Lastly, they are likely expert users. For these reasons, they do not represent well the common worker in the field, who has not yet been exposed to the system.

- Having a designer explain the user interface imposes the designer's mental model on users. But in production, users will not have designers standing over their shoulders, to explain the interface and guide them in their tasks.

We use a different approach, which on other engagements has resulted in systems that users feel are more usable. To address the first concern above, we conduct official usability tests with typical State and County workers, who have not yet been exposed to the system. These people better represent the ultimate user community. This gives us objective feedback, as these people have not been prejudiced by earlier design work.

Second, we use the following method to conduct usability tests:

1. A facilitator presents scenarios and associated low-fidelity diagrams. (Later in Software Architecture Design, we may begin using the high-fidelity diagrams.) We reuse our analysis use cases to drive our testing scenarios for usability. Recall that use cases represent the interaction between actors (e.g., users) and the system. We derived these use cases from requirements; therefore our usability testing is based on real-world requirements.

2. The facilitator then asks the user to accomplish the task. For illustration, the task might be a use case for verifying the results from a locate request.

3. The facilitator encourages the user to talk openly about his thought process. The user might say, "OK, so I just got an alert about a new address. I press this button, to take me to the address...now that I'm looking at the address...hold on a second, hmmm...OK, now I press the confirmation button, over there. I guess now I just return to my work list…yes that's it, on the main screen."

4. Someone in the role of the system responds to user actions – just as the system would. For example, if the user followed the process correctly, the person acting as the system would move the next page in front of the user. If the user made a mistake, the person acting as the system would politely "beep" and place a page representing the error message in front of the user. A fundamental point is that nobody jumps in to help a struggling user, to explain the interface, or to justify design decisions. That would force the designer's view of the interface onto the user. Instead, we use testing to discern the user's mental model. This helps us measure how intuitive the system is – that is, how usable it is for new users.

5. An observer records the user's actions and thoughts. This includes how the user reacts, how he hesitates, how he misses concealed menu items, and how he presses wrong buttons and inputs wrong data. We use these valuable observations as input in later discussions on improving the interface. Such discussions might include comments like "I had no idea multiple users did not figure out that when a new address is received from locate, an address verification letter must be generated. We should automatically take users to the forms generation page, so they can create that letter."

In this manner, **we use testing to understand the user's mental model, rather than forcing the user to understand the designer's mental model.** This naturally results in a more usable and intuitive user interface.

Large commercial software firms approach usability in a similar way. We believe that our robust methods will significantly improve the quality – and user acceptance – of the CCSAS CSE application.

Accenture brings to the alliance its intimate understanding of usability models it delivered on other large, public sector projects, which have used Internet technologies. This gives the alliance a head start with models driven by substantial input from government employees. We bring to the table new and proven usability ideas – but we also take time to test them with our users.

3C.12.3.3 Proven Work Products

In addition to these principles, analyzing software requirements requires secondary work products to aid in requirements gathering. Standardized diagrams and other pictures are necessary to describe the desired functionality of any complex software item. We go

beyond the typical approach of documenting requirements as simple text in a spreadsheet. We produce models that more thoroughly depict the complete requirements.
We continue to analyze requirements with use cases and the Rational Unified Process Requirements Workflow. Only now, we develop detailed use cases to supplement the high-level use case model of System Requirements Analysis. We also introduce a new work product: low-fidelity user interfaces.

The following paragraphs describe these work products.

3C.12.3.3.1 Use Cases

As introduced in Section 3C.12.1 System Requirements Analysis, use cases define the interactions between external actors (i.e. end users or other systems) and the CCSAS CSE application.

In addition to defining flows and software boundaries, use cases provide:

- A benchmark for project planning. A project manager can look at a use case, identify the scenarios to be delivered in an increment, and then build a work plan from there. In this manner the work plan tasks tie back to specific requirements.
- An input to testing. In this capacity, they can serve as the basis for developing more detailed test cases, which are traceable back to requirements.
- A useful communication tool for software developers and users. Both groups can review the use cases and understand what the system should do. To achieve a common language between these groups, we write use cases and use case scenarios in plain and understandable language.

We write detailed use cases, based on detailed functional requirements, during software analysis. The term "use case" often refers to a composite work product that consists of:

- **Use case diagrams,** which were discussed in Section 3C.12.1 System Requirements Analysis.
- A particular path through a use case is a **scenario:** a step-by-step sequence of message exchanges among the actors and the associated use case. A scenario specifies one sequence of interactions with one particular outcome. A use case can have multiple scenarios.
- **A use case realization diagram** illustrates a complex and related set of scenarios. We introduce these in Section 3C.12.4 Software Architecture Design.
- A **use case description** is simply the combination of various scenarios (i.e., paths) for a single use case.
- An **actor definition** provides a brief description about the actor, in text format.

The use case model describes how requirements come together in the software to achieve business goals.

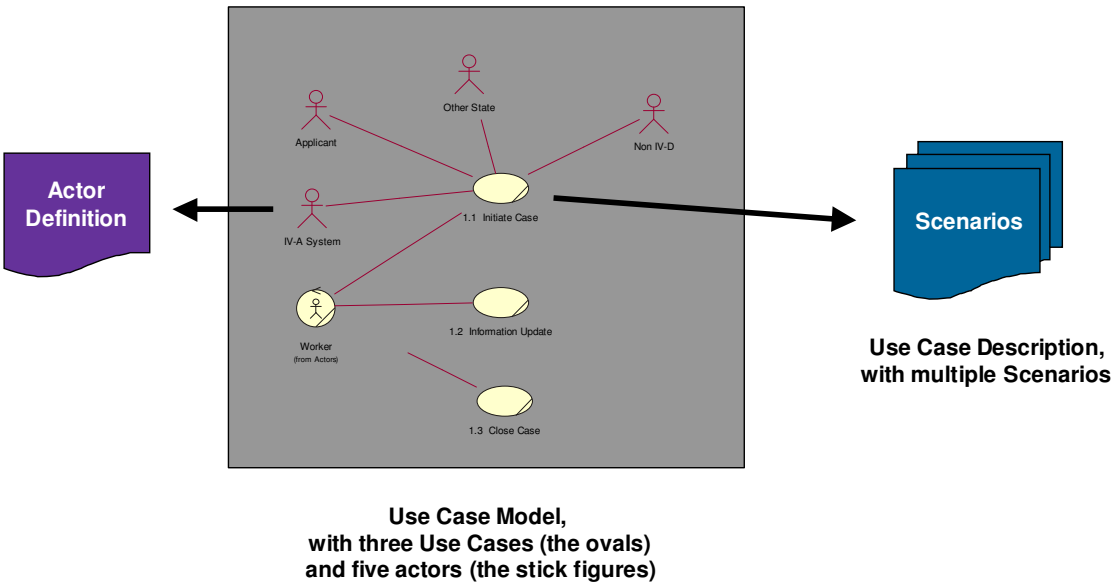


Figure 3C.12.3-1 Use Case Model – *A use case model includes use cases, use case descriptions, scenarios, and actor definitions.*

We continue to use Rational Rose for developing the use case model in standard UML notation.

3C.12.3.3.2 Low-Fidelity User Interface Diagrams

When describing functional requirements, users often have a user interface design already in mind. Their mental design might be based on experience with legacy systems or on their vision of how the new system should work. Creating a user interface is not part of requirements analysis; it is part of design. However, providing a simple, inexpensive example for users during analysis can help drive out more detailed requirements.

We model the user interface with low-fidelity user interface diagrams. This proven technique has backing in the Human-Computer Interface (HCI) community. We construct “low-fi’s” with pieces of construction paper, pencil, sticky notes, and tape. Inexpensive to create, these diagrams easily accommodate change. For illustration, a user can easily move around a sticky note containing a hand-drawn list box, in the middle of a low-fi review meeting.

These diagrams also accommodate change for more subtle reasons, based on psychology. Our experience – supported by formal usability studies – has shown that people suggest more substantial changes to diagrams that appear unfinished. We purposefully construct diagrams in this way, to invite feedback. We also use low-fi’s to focus users on the right level of detail. This includes the conversation flow for user interfaces and the high-level business processes. Were we to use computer-rendered diagrams instead, the diagrams would distract users from the important usability aspects. Users might focus on misaligned entry fields or improperly sized buttons, for

instance. This is natural: diagrams with a finalized look encourage people to concentrate on finalized details.

That said, our hand-drawn diagrams do follow, in spirit, the high-level usability standards for the system architecture. For example, the low-fi's reflect if the system usability model dictates the use of HTML tabs for sub-functions on the same page. In this manner, the detailed user interfaces that we create later, in Software Architecture Design, will meet our usability standards.

Low fidelity diagrams are low-cost but high-value ways to clarify requirements from users.

Low-fi's are analogous to white-board designs. Both are hand drawn, both are easily erased and modified, and both quickly result in valuable feedback – at the right level of detail. Just as we use white-boarding in the early stages of component design, we draw low-fi's in the early stage of user interface design. The alliance has used low-fi's successfully on other engagements, resulting in higher quality user interfaces. We believe that low-fi's differentiate our approach from that of competitors.

We only use low-fi's as a tool to analyze and clarify existing requirements. We emphasize that project staff and users not treat the low-fi's as the official design at this point. We actually design the user interface in the next stage, Software Architecture Design. Our hand-drawn diagrams do follow, in spirit, the usability standards initially defined for the system architecture. For example, the low-fi's reflect if the system usability model dictates that HTML tabs be used for sub-functions on the same page. In this manner, the detailed user interfaces that we create in Software Architecture Design will meet our usability standards.

We use the processes defined in Section 3C.4 Configuration Management for our low-fi's. We scan approved versions of the paper-based diagrams and check them into our version management repository. This repository stores not only code modules, but also more generic electronic documentation like scanned, low-fi images. We assign each scanned file a unique id based on the collection of windows that file represents.

Low-fi's are a key part of our approach. They help us arrive at a comprehensive, consistent, and complete set of functional and usability (i.e., behavioral) requirements.

3C.12.3.4 Proven Process

The decision makers from System Requirements Analysis carry on in Software Requirements Analysis. This gives continuity, avoids re-training, and supports our aggressive schedule.

Our robust software analysis includes the following steps:

1. Conduct Team Training (for new skills relevant to software analysis)
2. Analyze Software Requirements
3. Prioritize Software Requirements

4. Create the IEEE-Based Software Requirements Specification (SRS)
5. Informally Review SRS
6. Begin Software Architecture Design
7. Acceptance of SRS

After following these steps for Version 1, we repeat them for Version 2. Version 2 software builds on Version 1 software and therefore builds on the Version 1 SRS. We do not expect many changes to work products created for Version 1 during Version 2 process steps. We handle those changes that do arise in a controlled manner, using our processes described in Section 3C.4 Configuration Management.

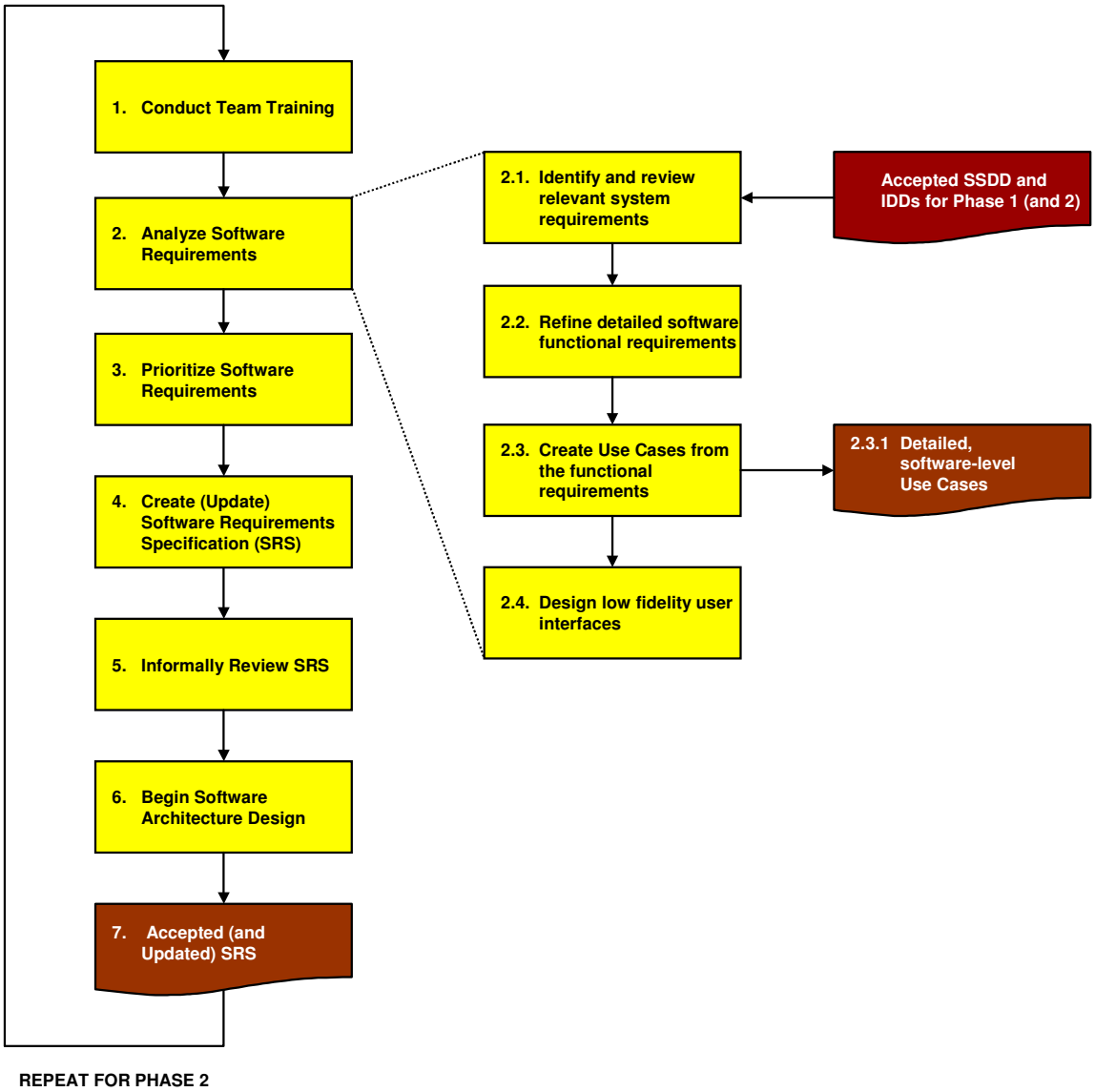


Figure 3C.12.3-2 Process Steps – *It's more than requirements gathering – our robust process analyzes the requirements.*

The following paragraphs describe these steps.

3C.12.3.4.1 Conduct Team Training

This step is similar to the training step in Section 3C.12.1 System Requirements Analysis. But this step differs in the tools and techniques demonstrated in the training sessions.

As with System Requirements Analysis, the formal training for the Software Requirements Analysis team aims to improve:

- The efficiency of the analysis process
- The quality of and confidence in the requirements

The State and alliance team members who analyze requirements need to understand the various analysis tools and practices. We conduct training sessions for their various roles:

- Requirements Meeting Facilitator
- Use Case Author
- Low Fidelity User Interface Designer
- Low Fidelity User Interface Meeting Facilitator

We conduct one training session, lasting approximately a half to a full day, for each of these roles.

The alliance and CCSAS CSE project staff will schedule these one-time, highly specialized sessions. This training is specific to the Requirements phase of the base contract. The knowledge gained is not needed for maintenance of the system. Therefore, it is not a part of the formal Knowledge Management Plan. Additional training requests will be addressed using the Change Request Management process. Any reproducible materials used in this training will be given to the State.

3C.12.3.4.2 Analyze Software Requirements

To analyze software requirements, we:
1. Identify and review the relevant system requirements
2. Refine detailed software functional requirements
3. Create use cases from the functional requirements
4. Design low fidelity user interfaces

The following paragraphs describe these steps.

3C.12.3.4.2.1 Identify and review the relevant system requirements

We have already identified high-level software components in the System/Subsystem Design Description (SSDD), based on system requirements. Each software component

derives from one to many system requirements (functional and/or technical). We review these requirements stored in DOORS to identify potential issues and questions relating to the software.

3C.12.3.4.2.2 Refine detailed software functional requirements

For requirements to support the subsequent design process, they must be explicit and testable. Many functional requirements gathered as part of the Business Analysis Approach and documented in the Business Requirements List at contract award likely do not provide enough detail. We have already categorized each previously gathered functional requirement as "summary" or "detailed" according to Section 3C.12.1 System Requirements Analysis. We now:

- Create detailed requirements for each summary functional requirement that lacks detailed requirements
- Refine existing, detailed functional requirements

We do not expand the scope of the requirements. Rather, we clarify the high-level or vague requirements to develop a stronger Software Architecture Design. We derive new detailed requirements from existing summary requirements. In this manner, we avoid increasing scope.[17]

System Analysis has already detailed technical requirements.

We write detailed requirements with the "testability" litmus test in mind. This enables the test team to create test conditions that explicitly trace back to their originating detailed requirements. See Section 3C.12.9 Test Management. The litmus test helps confirm that the requirements are sufficiently expressed – that they will be valuable for Software Architecture Design.

3C.12.3.4.2.3 Create use cases based on the functional requirements

Next, we more formally analyze the functional requirements. The functional requirements have been enumerated in the requirements tool, essentially in a list. We have a hierarchical relationship – a vertical link – between summary and detailed requirements. But we lack a horizontal link that relates summary requirements with each other or detailed requirements with each other. From this perspective, the requirements are loosely affiliated, making them very cumbersome to analyze. Moreover, it is nearly impossible to discern the order in which business events occur to achieve system goals.

To solve this, the analysis team models the summary and detailed functional requirements with detailed use case diagrams.

[17] The point is that this activity does not set out to increase scope. It refines and documents existing scope. However, we do not preclude scope changes to requirements. We handle such changes in accordance with Section 3A.4 Change Request Management.

Use cases created now differ from the high-level use cases of System Requirements Analysis. They simply differ in level of detail. These new use cases depict discrete interactions between an actor (e.g., user) and the software. They describe the detailed actions for achieving some business goal. We need more detailed, software-level use cases to properly analyze the functional software requirements. These detailed use cases provide the necessary context to create a Software Architecture Design.

Software use cases describe *detailed* user interactions with the software.

3C.12.3.4.2.4 Design Low Fidelity User Interfaces

As described earlier, low fidelity user interfaces are simple diagrams of possible graphical user interfaces. We use these diagrams to quickly analyze a collection of suggested requirements and confirm that they represent what the user community intends. We test their usability with the decision makers.

We use these pencil-down interfaces only to help confirm requirements. They should not be treated as formal deliverables. Instead, formal user interface design occurs during Software Architecture Design. At that time, we model the user interface with more detail and focus.

3C.12.3.4.3 Prioritize Software Requirements

This step mirrors the step that prioritizes requirements for System Requirements Analysis, in Section 3C.12.1. The only difference is that we now prioritize software requirements rather than system requirements.

The analysis team works with decision makers to prioritize the requirements, thereby giving requirements context. Priorities provide input to a variety of tasks, such as identifying opportunities for prototypes or assigning the build-out of requirements to work plan increments.

Individual requirements, in their raw form, do not lend themselves to prioritization. They describe a single step to reaching a business goal. Use cases, on the other hand, organize functional requirements by expressing relationships and order. Use cases are easier to prioritize as they describe a whole business goal. So the team assigns priorities to use cases.

We treat technical requirements differently from functional requirements for prioritization. This is because a technical requirement does not describe a step in reaching a business goal. Rather, this type of requirement describes qualities of service for any number of business goals.

We prioritize detailed use cases and against a number of criteria, including:

- Business value
- Complexity
- Size
- Impact
- Level of understanding
- Likelihood for change

Using various criteria makes the prioritization useful. For example, we might design a use case with high complexity and low level of understanding earlier than other use cases. This helps identify issues, uncover dependencies, and gain a better understanding of associated work plan tasks. As another example, we might schedule a use case with a high likelihood for change later in the cycle to let the use case stabilize.

Detailed use cases contain base and alternative flows, and each path might have a different priority. We prioritize the base and alternative flows of any given use cases separately, to achieve the most accurate assessment.

Lastly, we document priorities either in DOORS or in the work plan.[18]

3C.12.3.4.4 Create the IEEE-Based Software Requirements Specification (SRS)

The Software Requirements Specification defines the requirements to be met by the software. This document is explained in annex F.2.4 of the IEEE J-STD-016-1995 standard.

This document refers to the following work products:

- **Functional Requirements** – A list of summary and detailed functional requirements stored in the DOORS repository.
- **Use Cases** – A collection of the business functions and rules of the software described as a series of detailed interactions between actors and the software. Functional requirements should be represented within a use case.
- **Low Fidelity User Interface Diagrams** – Simple (pencil and paper) illustrations of the proposed user interface. These drive out detailed functional software requirements relating to user interfaces. We store and version scanned images of these diagrams in a repository.

[18] Priorities are primarily an input for work planning. As the alliance creates detailed procedures for work planning, it determines whether storing these requirements in DOORS or in the work plan makes the most sense.

3C.12.3.4.5 Informally Review Software Requirements Specification (SRS)

This step mirrors the step for informal review in System Requirements Analysis.

Decision makers review the SRS on behalf of the CCSAS Project Staff and users that they represent. This informal review does not aim to eliminate remaining issues or questions. Instead, this review confirms that the analysis team understands the highest priority requirements and use cases. Then, the development team can begin Software Architecture Design to validate the requirements.

3C.12.3.4.6 Begin Software Architecture Design

This step mirrors the step for beginning architecture design in System Requirements Analysis.

We begin Software Architecture Design before official acceptance of the SRS. Overlapping this design with Software Requirements Analysis is consistent with IEEE. This allows the team to validate that the software requirements are consistent and feasible. We consider initial Software Architecture Design to be part of Software Requirements Analysis. Feedback from the design will affect the requirements.

3C.12.3.4.7 Acceptance of Software Requirements Specification (SRS)

The goal of Software Requirements Analysis is the delivery of an SRS containing requirements that are:

- Traceable to the system requirements and system design
- Externally consistent with the system requirements
- Internally consistent
- Testable
- Feasible with the software architectural design
- Feasible with planned operation and maintenance of the software

The decision makers use these criteria to participate in the Software Requirements Review.[19] We then submit the SRS for formal deliverable walkthrough. After we have identified and addressed major issues from this walkthrough, then:

- The SRS enters the formal CDL acceptance process agreed upon by the alliance and CCSAS project staff.
- The accepted SRS is baselined into the configuration management (CM) process. This does not preclude future changes to the SRS. We believe that requirements evolve throughout the project's lifetime, such as from legislative changes. We apply formal CM procedures to evaluate proposed changes for their impact on the project.

[19] See Section 3C.8 Technical Reviews.

3C.12.3.5 Conclusion

Software Requirements Analysis resembles System Requirements Analysis. We use similar techniques, tools, and process steps. But it differs in focus: we now look at software components and user interfaces, to define what the software does or provides. We also emphasize functional requirements, which have a greater role in Software Architecture Design.

We create new work products that add priority and structure around previously gathered requirements.

We begin looking at the user interface model. We use an innovative approach to usability testing, which focuses on the user's mental model, rather than that of the designer. We also introduce low-fidelity user interfaces, a new analysis work product that helps drive out functional and behavioral requirements.

From an organization perspective, our thorough software analysis includes:

- Trained analysis team members, to moderate discussions with decision makers
- A cross-section of relevant decision makers, not just those with business knowledge
- Functional Subject Matter Experts provided by the development team, for enhancing the scope of knowledge and experience of the user community

Lastly, we overlap Software Architecture Design with Software Requirements Analysis. The alliance's work products, coupled with the overlap, validate detailed functional requirements. This, in turn, validates that we can build usable CCSAS CSE software.

3C.12.4 Software Architecture Design

The Software Architecture Design defines how the software fulfills its requirements. This phase validates the accuracy and feasibility of the software requirements. During design, we apply proven component principles and leverage production-ready architecture assets. We deliver a software architecture that meets not only SCP architecture best practices, but also those of the alliance.

3C.12.4.1 Introduction

Section 3C.12.2 System Architecture Design explains the process for defining the CSE software items. But that stage did not address details of the software items, like their graphical user interface or internal structure. It is in this stage, Software Architecture Design, that we address the details for each software item so defined.

Software Architecture Design addresses the internal workings of the software. We focus on defining *how* the software addresses its assigned requirements. In addition, Software Architecture Design takes the documented user interface requirements to the next level of detail. Based on the low fidelity user interface diagrams created during requirements analysis, we produce high fidelity diagrams. These diagrams translate the user interface requirements into a more detailed illustration of the users needs. We also develop prototypes for potential software users. The users interact with these prototypes in an effort to validate the requirements' accuracy and feasibility.

Software architecture builds out the internals of the components defined in the system architecture.

3C.12.4.2 Proven Principles

As with system architecture, a consistent software architecture must be designed with a set of principles in mind. We continue to use the applicable principles addressed in System Architecture Design, in particular:

- Architecture Layering
- "Best of Breed" Design Components
- Requirements Validation with the Architecture Design (only this time we use the *Software* Architecture to validate the *software* requirements)

Section 3C.12.2 System Architecture Design has already explained these principles. In addition, for Software Architecture Design, we introduce the following new principles:

- Component-Based Design
- Proven Practices for Component Architecture
- Reusable Architecture Assets for Improving Maintainability

The following paragraphs describe these new principles.

3C.12.4.2.1 Component-Based Design

We use component technologies and design approaches for the CSE system architecture.

Components have been around for a long time. The wheels on an ancient Roman chariot were components. When the local chariot maker invented a new wheel (one that promised greater speeds and improved reliability on a wider variety of terrain), chariot owners would replace their worn-out, inefficient and out-dated wheels with the new ones, but only if the new ones offered, at a minimum, the same function (i.e., rolling) through the same interface (i.e., the connection between the wheel and the chariot).
Today, components are used to build everything from cars to computers. In electronics, for example, they have led to the proliferation of product features, disposability, miniaturization, product selection, price reduction and standard interfaces – all good for the consumer.

Throughout the industry, the words "software component" are used broadly and often loosely. Components come in a wide variety of shapes and sizes. For example: JavaBeans, CORBA components, ActiveX controls and DCOM objects. And more generically: application, architecture, development, engineering, web, server, and business components.

Many industry experts have attempted to define "component." Unfortunately, many of these definitions are too abstract, too academic, or too specialized. We suggest the following definition, which aims to be simple and concise:

> **A component is an encapsulated, reusable piece of software that fulfills a clear function through standard, well-defined interfaces.**

“Interfaces” in component terminology refers to Application Programming Interfaces (APIs)[20] rather than user interfaces.

Engagement experience has shown that it is quite common for people to view components from different perspectives. Some people, typically designers, take a logical perspective. They view components

Component-oriented design principles are fundamental to the software architecture.

[20] An API is a set of related functions (e.g., subroutines) that other programs can call.

as a means for modeling real-world concepts in the business domain. These are *Business Components.* Other people, typically developers, take a physical perspective. They view components as independent pieces of software, or application building blocks, that implement those real-world business concepts. These are *Partitioned Business Components.* Developers also emphasize that Partitioned Business Components can be built from other independent pieces of software that provide functionality that is generally useful across a wide range of applications. These are *Engineering Components.*

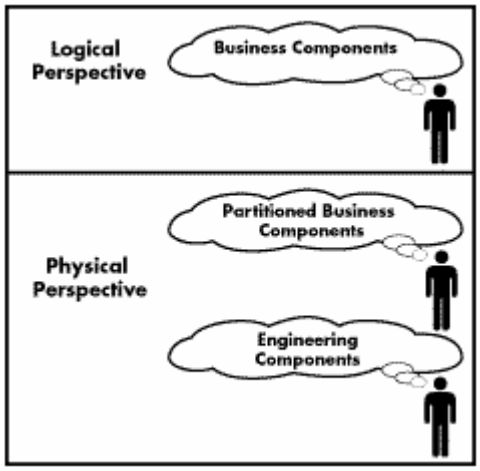


Figure 3C.12.4-1 Different Component Perspectives – *For System Architecture, we think about business components. We refine these into Partitioned Business Components for Software Architecture. The architecture team uses Engineering Components for reusable technical modules.*

3C.12.4.2.1.1 Business Components

Business Components represent real-world concepts in the business domain. They encapsulate the information around those concepts including name, purpose, knowledge, behavior and other intelligence. Section 4A.3 CSE System Conceptual Design introduces high-level Business Components such as: Case Initiation and Update, Establishment, Enforcement, Locate, Obligation Management, Collections, Distribution, Disbursement, Program Monitoring, IV-D Fund Management, and Customer Service.

One might think of a Business Component as a depiction or portrait of a particular business concept. As a whole, the Business Component Model is a depiction of the entire business. Although this begins the process of defining the application architecture for a set of desired business capabilities, the applicability of the Business Component Model extends beyond application building.

3C.12.4.2.1.2 Partitioned Business Components

Whereas Business Components model real-world concepts in the business domain, Partitioned Business Components implement those concepts in a particular

environment. They are the physical building blocks used to assemble applications. As independent pieces of software, they encapsulate business data and operations, and they fulfill distinct business services through well-defined interfaces.

We transform Business Components into Partitioned Business Components based on the realities of the technical environment: distribution requirements, legacy integration, performance constraints, existing components and more. Business Components represent the logical model, whereas Partitioned Business Components represent the physical model. That is, we build out Partitioned Business Components as executable software.
For example, the Case Initiation and Update Business Component can be broken into various Partitioned Business components. Some examples might include:

- Case Construct – e.g., what are the rules that define case construct?
- Case Transfer – e.g., how does CSE handle ownership of a case when in multiple counties?
- Case Assignment – e.g., who is assigned to the case as it moves through its lifecycle?

3C.12.4.2.1.3 Engineering Components

Frequently thought of as "all other components," Engineering Components are independent pieces of technical software that provide functionality useful across a range of applications. They come in different shapes and sizes, and they are typically packaged as black box capabilities with well-defined interfaces.

They are the physical building blocks used in the assembly of Partitioned Business Components. Examples include: a workflow engine, a JavaBean that encapsulates a reusable concept like address or monetary unit, a complex widget that allows users to edit a list of order lines, a group of objects responsible for persistence, a JavaBean that sorts a collection of objects, and a simple list box coded as an ActiveX control.

3C.12.4.2.1.4 Component Characteristics

The following characteristics describe components in more detail. A component need not demonstrate all of these characteristics to be considered a component. A checkmark means that a particular characteristic applies to Business Components or to Physical Components (i.e., Partitioned Business Components and Engineering Components). For example, components, in general, should be encapsulated, whereas only physical components are executable (i.e., can be run as software).

Business Components	Physical Components	Characteristic	Description
✓	✓	Encapsulated	A good component is a black box with predictable, well defined interfaces. Requesters of its services know what it does, but not how it does them. Its internal workings are both hidden and isolated; they can be implemented using any technology.

Business Components	Physical Components	Characteristic	Description
✓	✓	Nearly independent	A component should be loosely coupled to other components. In other words, the dependency between any two components should be minimal. If a component changes, the impact on other components should be minimal, and vice-versa.
✓	✓	Highly cohesive	Cohesion is the "state of sticking together tightly." A component exhibits this characteristic if its purpose is clearly defined, unmistakable and precisely focused. All of its services should contribute to this purpose.
✓	✓	Trusted and marketable	An existing component should be proven in a prior implementation, and it should perform without undocumented side-effects. A new component should strive for this characteristic. It should provide functionality that is desirable to consumers other than the original developers.
✓	✓	Reusable	After an organization has designed, built, or purchased a component, anyone can use it to perform its specified services. This promotes the assembly of applications from components. To achieve this goal, components must be developed to be used in unpredictable combinations.
✓	✓	Replaceable	It should be simple to replace a component as long as the new component offers, at a minimum, the same set of services through the same interfaces. Most likely though, the new component will probably offer an expanded set of services.
	✓	Executable	A component can be run within a component environment (e.g., application server), without needing access to its source code.
	✓	Distributable	Components can be distributed for execution using standards for component interfaces and middleware services (e.g., COM, JavaBeans, and CORBA).
	✓	Scalable	A component can be configured to execute on any number of servers. Consequently, an application can adapt more readily to changing transaction volumes.
	✓	Interoperable	As long as a component adheres to standards for component interfaces and middleware services, it should be possible to request the component's services from any platform.
	✓	Self-describing	A component should be able to describe its public interfaces, any properties that are customizable and the events that it generates.

Figure 3C.12.4-2 Component Characteristics - *Designing good components requires mastery of several different characteristics.*

A “component-based design,” then, decomposes complex system software into components. These components are loosely-coupled but well integrated. In other words, they can accomplish a lot of work on their own, but they work well together when necessary. The sub-section on Work Products gives an example of a high-level component model.

For Software Architecture Design, we continue to use the Rational Unified Process (RUP). In this manner, we follow a component-based design approach.

The alliance has years of experience building large-scale, component-based solutions in a variety of industries using a variety of technologies. As well, IBM brings to the alliance market-leading software for building and deploying component-based systems on an open, J2EE platform.

3C.12.4.2.2 Proven Practices for Component Architectures

Building a flexible, robust component design is more than deciding to use an object-oriented language that supports inheritance and information hiding. Without applying proven component principles, many of the same problems that plague procedural language systems can sneak into component-based systems.

The alliance brings years of experience building strong component-based architectures. This experience is embodied in a set of design principles and patterns that increase software maintainability. These include:

- Organize application code into business components
- Separate process components from entity components
- Wrap legacy, packaged, or eAI calls as business components
- Support loosely coupled communication between components
- Define transaction boundaries at the business component level
- Separate the presentation layer from the underlying control layer
- Separate the control layer from the underlying model
- Define the control layer as a series of nodes in a process
- Separate data access logic from business logic
- Use a role-based security mechanism

The following paragraphs describe these practices.

3C.12.4.2.2.1 Organize application code into business components

We structure the software's business logic as large-grained, reusable business components. We design these components primarily using service-based interfaces. In a service-based interface the client application makes its request in a single message to the component, the component responds in a single return message, and the component releases any state for that client. Releasing this state improves both throughput and scalability. Because service-based interfaces involve a more straightforward communication, they tend to be more reusable. The J2EE specification directly supports such components, which are referred to as "stateless".
Where appropriate, we design these interfaces to be document-based. By transferring data using the Extensible Markup Language (XML) format, the interfaces are more reusable by future, external systems.

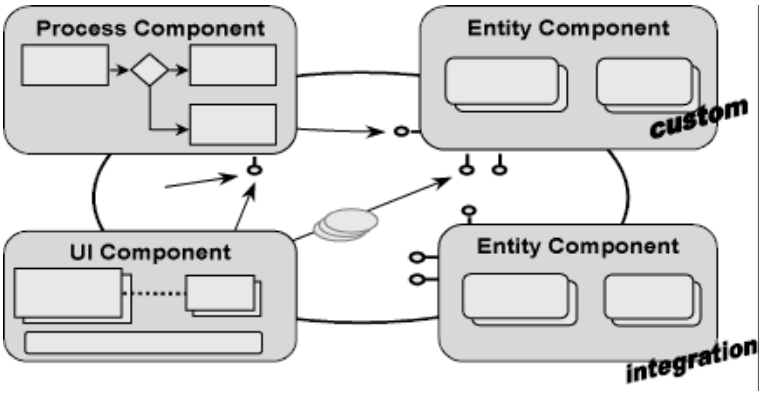


Figure 3C.12.4-3 Organize Application Code Into Business Components - *Components hide their internal details from the outside world, from other components.*

3C.12.4.2.2.2 Separate process components from entity components

A process component defines a set of business processes and rules. An entity component, on the other hand, defines the business data (and its relationships to other data) that the business rules apply to. We keep these two types of components separate and represent them differently, using standard J2EE EJB practices.

By separating process and entity components, the business rules do not have to be concerned with the business data's source or internal structure. The entity component is responsible for obtaining data, maintaining its relationships, and saving the data to a persistent storage device. The process component only knows how to read from and write to an entity component. We achieve greater maintainability be keeping these two very different responsibilities separate in the design and code.

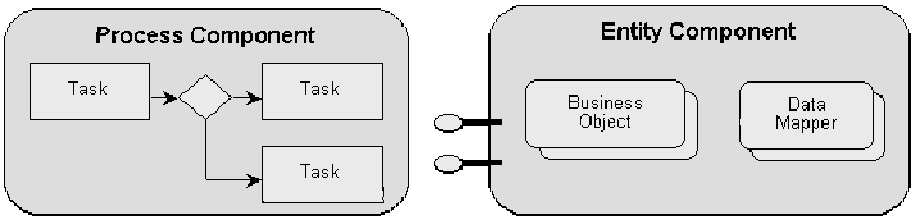


Figure 3C.12.4-4 Separate Process and Entity Components - *Process components embody business policies and activities, whereas entity components represent business data.*

This principle should not be viewed as a departure from basic object-oriented principles. We still follow the fundamental principles of encapsulation and data hiding. That is, our object classes are more than just data structures – they wrap methods around their data. (Section 3C.12.5 Software Detailed Design defines class modules.) Components, whether entity or process, contain multiple classes. In other words, we build components using object-oriented techniques.

The point of this principle is that we organize components to separate out logic representing business entities from logic representing business policies and rules. This supports business process engineering: we can more easily maintain high-level processes, like “locate,” independently of the entities, like “case” and “person,” that they act upon.

3C.12.4.2.2.3 Wrap legacy, packaged, or eAI calls as business components

We insulate the business logic from the technical details of integrating with external software or systems. To make the communication from business logic consistent, these technical details are represented as business components. The process components make calls to these components like any component. As with entity components, the separation of concern between business logic and integration logic improves software maintainability.

Additionally, we design the external interfaces to use XML where appropriate. By serializing data in a standard format, we gain greater flexibility and potential for reuse.

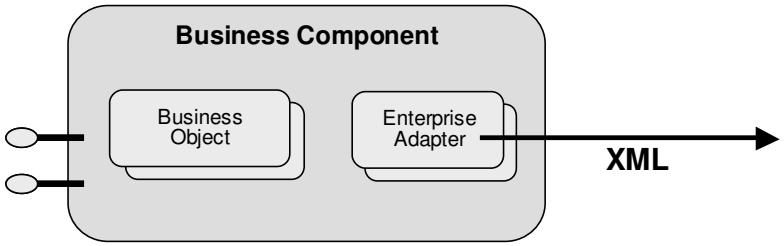


Figure 3C.12.4-5 Wrap Integration Calls as Components - *Do not expose legacy, packaged, or eAI calls to the presentation or control layers.*

See Section 3C.12.8 System Integration for details on XML.

3C.12.4.2.2.4 Support loosely coupled communication between components

Complex component-oriented software will contain a significant number of components. These components need to interact with each other to complete their business goals. The difficulty here is designing highly cohesive components without creating unreasonable coupling. In order to avoid this design pitfall, the alliance attempts to design component interfaces using publish-subscribe and other asynchronous mechanisms.

In a publish-subscribe interface, a component (the publisher) triggers a particular type of event. Other components that have registered an interest in this same type of event (subscribers) will receive it. The publisher has no knowledge of who subscribes to the event. This loose coupling means that subscriptions can be updated without impacting the publisher.

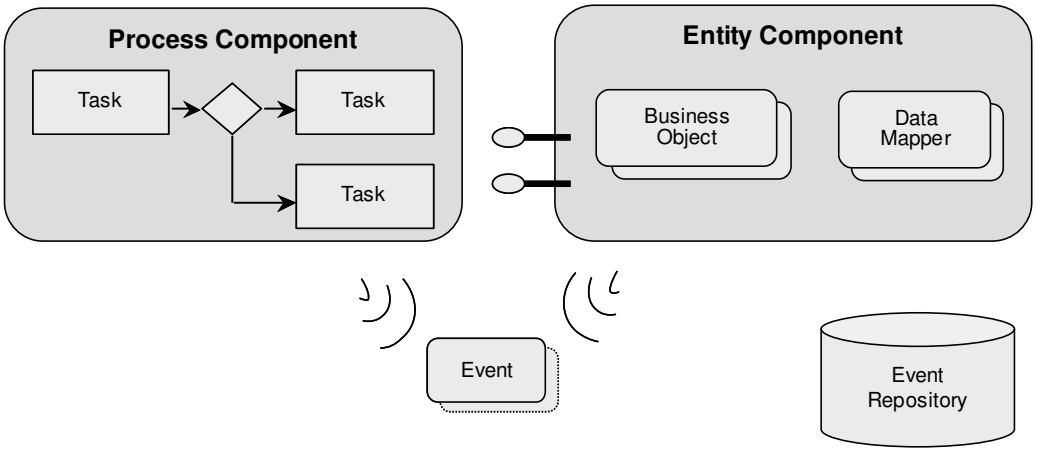


Figure 3C.12.4-6 Loose Coupling - *Use loosely coupled communication like publish and subscribe where appropriate.*

This type of interface can be tied into the messaging layer of the system so that events can seamlessly be published both inside and outside the bounds of the software.

3C.12.4.2.2.5 Define transaction boundaries at the business component level

We define transactions as part of the component's interface. When a component service is invoked, a new transaction begins automatically. The technical complexity of the transaction is hidden in the component so the caller does not have to deal with it. The J2EE EJB specification allows for this type of transaction handling. The alliance applies experience with EJB transaction management to build this principle into designs.

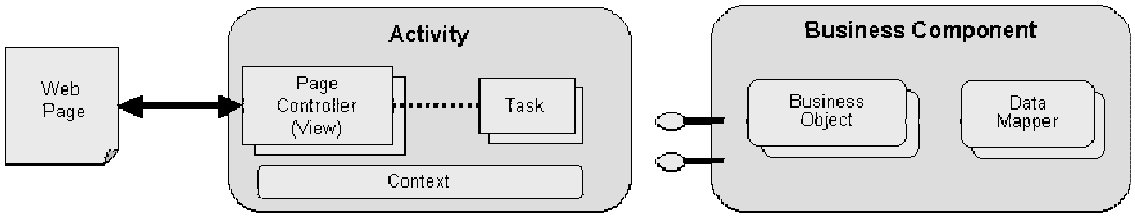


Figure 3C.12.4-7 Transaction Boundaries - *Let business components handle transactions.*

3C.12.4.2.2.6 Separate the presentation layer from the underlying control layer

In Section 3C.12.2 System Architecture Design, we describe layering principles to separate out presentation logic. The presentation layer can be further broken down into the actual presentation logic and the presentation control logic. The presentation logic is only responsible for how information is displayed to the user. The control logic, on the other hand, handles user actions, such as when a button or link is clicked.

The alliance uses patterns that separate these two types of logic within the presentation layer. This separation allows for reuse of control logic across multiple presentation formats (like HTML and CTI). Also, data validation logic can be separated into the layer where it makes most sense. Invariant and presentation-specific rules are placed in the presentation logic where more complex or common validation rules are placed in the control logic. Again, this separation supports improved maintainability.

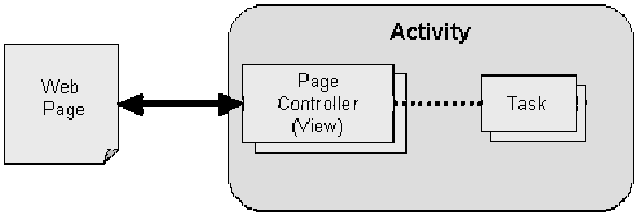


Figure 3C.12.4-8 Isolating Presentation - *Split the presentation layer into true presentation logic and underlying control logic.*

3C.12.4.2.2.7 Separate the control layer from the underlying model

Similar to the separation of presentation from control, the alliance also separates the control from the model being represented and manipulated. The model is the data representation of the presentation layer. The control layer handles the application of business rules to one or more business models. The model only knows about the data it is responsible for and is, therefore, easier to build and maintain. The model also becomes much more reusable across different control logic, further reducing the total development and maintenance effort.

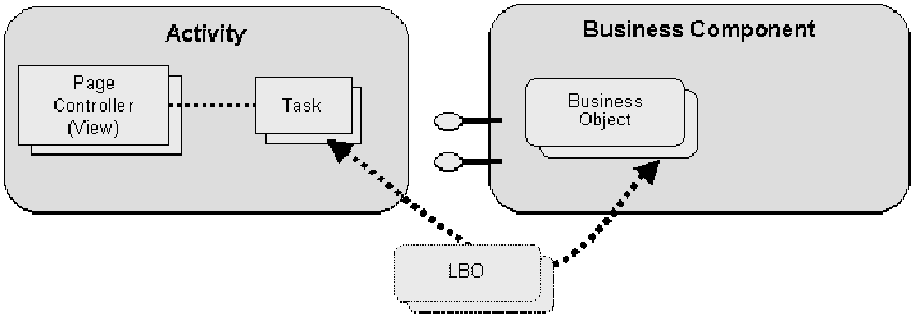


Figure 3C.12.4-9 Separating Control and Model - *Use MVC and pass light-weight business objects (LBOs) by "value" as opposed to by "reference."*

This separation of presentation, control, and model is the application of the widely-used Model-View-Controller (MVC) pattern. The alliance has a great deal of implementation experience with this pattern and has seen much success with it.

3C.12.4.2.2.8 Define the control layer as a series of nodes in a process

An activity in the control layer usually represents a single transaction. A transaction contains a series of discrete tasks that all must complete before the transaction itself is complete. We follow proven object-oriented design practices and represent these steps as task objects within an activity.

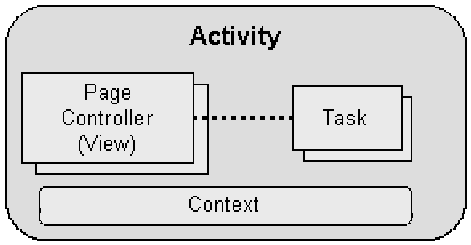


Figure 3C.12.4-10 An Activity, Its Tasks, and Their Shared Context - *We connect tasks in a loosely coupled fashion, both for flow and for sharing memory.*

Task objects are connected in a loosely coupled fashion. The activity controls the flow between tasks. Moreover, we use a loosely-coupled approach to storing state between tasks, with a separate context object. Because these two techniques reduce the dependencies between tasks, we can rearrange tasks to support flexible process flow. This allows for the addition or removal of tasks without requiring changes to other tasks. Flow and state logic is maintained by the activity, making the tasks more cohesive and reusable.

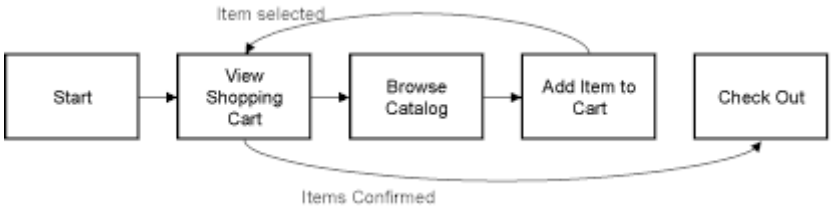


Figure 3C.12.4-11 Flexible Process Flow - *The decoupling of tasks better supports Business Process Reengineering (BPR).*

3C.12.4.2.2.9 Separate data access logic from business logic

Earlier, we proposed the concept of a business component to represent business entities. These business components provide interfaces to access business data. The business component is responsible for obtaining data from a persistent store, implementing business data rules, and storing changes back to the persistent store.

We separate the business data rules from the logic that stores to and retrieves from the database. This separation allows for changes to be made to the data access logic (e.g., database changes such as table or column name changes) without affecting the business data rules and relationships. Data can be accessed using a number of

standard mechanisms (SQL, stored procedures, eAI messaging) and the business rules remain unconcerned by it.

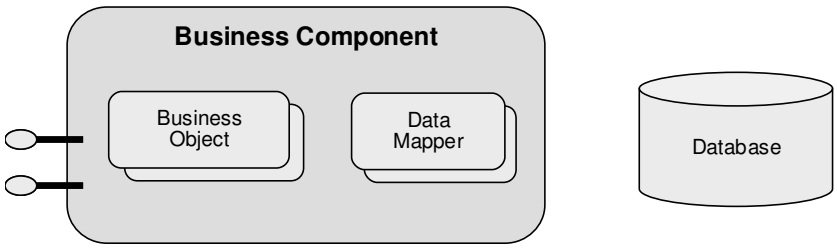


Figure 3C.12.4-12 Separating Data Access - *Data Mappers separate business logic from data access logic. Mappers define the relationship between a business object and a data source.*

This overarching principle was addressed in Section 3C.12.2 System Architecture Design. This design pattern is an implementation of that principle.

3C.12.4.2.2.10 Use a role-based security mechanism

CCSAS protects confidential information about child support obligations, financial transactions, court orders, etc. As such, CCSAS CSE requires robust security access logic.

We avoid weaving this logic throughout the application code by representing user roles as objects. After authenticating a user, an assigned role object is created and can be accessed by the software. Access to the role and user information is done at the control layer. The controller can query the role object to ascertain whether access can be granted to a particular function. This encapsulates the access control logic within the role object. This avoids the proliferation of complex if-then-else statements throughout the code.

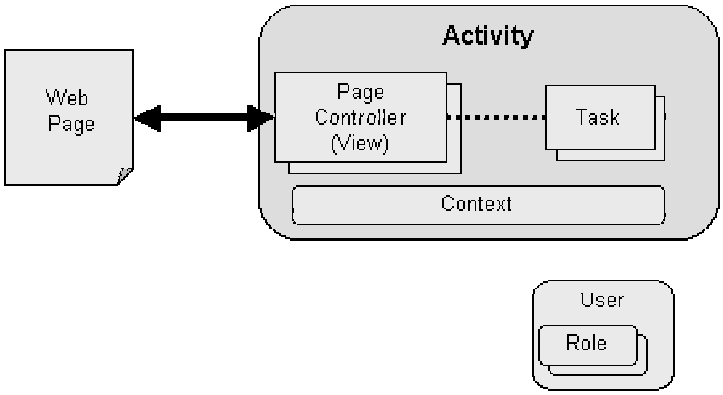


Figure 3C.12.4-13 Role-based Security - *Represent user roles as objects that can be passed around as part of a transaction.*

3C.12.4.2.3 Reusable Architecture Assets for Improving Maintainability

In addition to proven component design principles, the alliance brings proven architecture assets. Accenture's GRNDS[21] is an Internet and component-based application architecture, built for the Java 2 Enterprise Edition (J2EE) and Web Services platforms. GRNDS provides many common technical services, including but not limited to: logging, error handling, security, database access, state management, and web conversation flow. These services are loosely coupled and configurable, allowing architects to use the entire offering or select specific services.

GRNDS leverages well-established design patterns and extensive architecture experience gained on large-scale enterprise engagements. The GRNDS application model specifies what type of logic goes where, including standards and guidelines. It follows the architecture layering principles described in Section 3C.12.2 System Architecture Design.

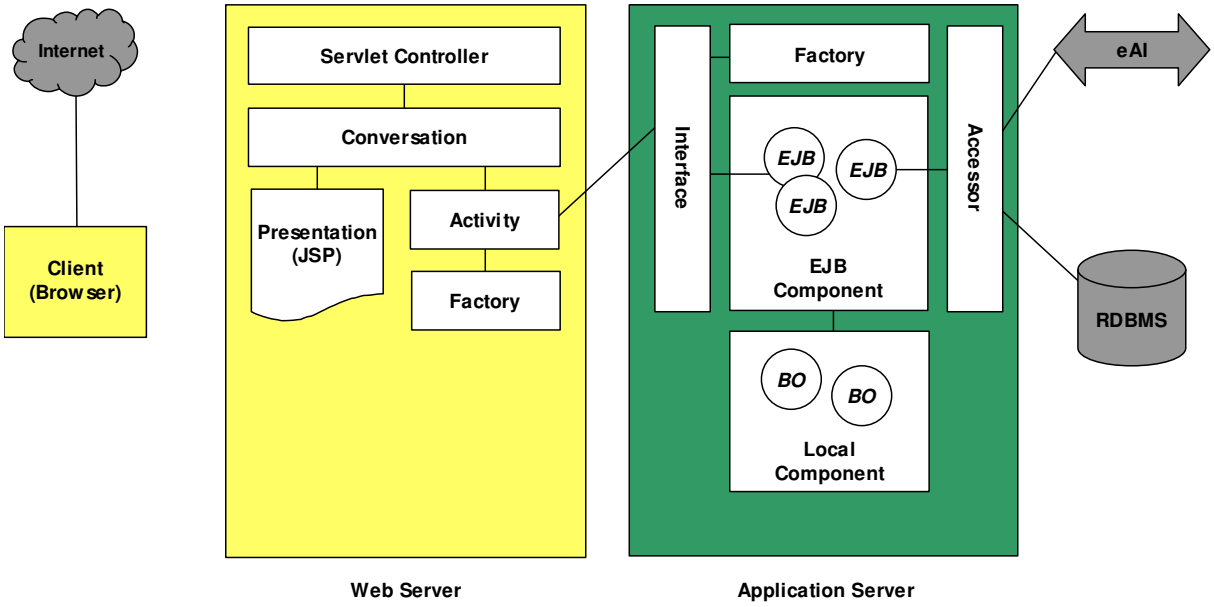


Figure 3C.12.4-14 GRNDS Application Model - *This model breaks down the software architecture for web and component applications, in a proven way.*

The GRNDS application model fills an important void. Out of the box, application servers and web servers do not provide significant guidance on the appropriate places for different types of logic. The GRNDS application model has shown, from use in production, that it is performant, robust, and maintainable.

[21] The GRNDS acronym stands for "General and Reusable Netcentric Delivery Solution."

GRNDS includes design documentation, user manuals, training workshops, and "cookbooks," which are straightforward "how do I…?" guides. Accenture supports GRNDS with a vibrant, internal open source[22] community for ongoing maintenance and development. This community includes discussion forums, mailing lists, and other mediums to tap into the wealth of knowledge in the GRNDS community. Accenture architects who work on the CCSAS Project will subscribe to these forums and mailing lists. Through those mediums, those architects will receive notifications of new releases of the GRNDS software. The CCSAS Project will evaluate the benefits as well as impacts of upgrading to a new release.

GRNDS has the following benefits:

- It improves the efficiency of application developers. Its Java and Web Service frameworks hide complex technical details from application developers. This allows developers to focus on designing and programming business functionality.

- It improves interoperability, portability, and scalability. A variety of large-scale engagements have production-tested GRNDS, with scalable results. GRNDS is interoperable, as well: it supports the industry J2EE standard and burgeoning Web Services standards. Lastly, GRNDS has been tested in a variety of J2EE application servers. This increases vendor independence.

- It reduces the total cost of maintenance. Because (1) GRNDS centralizes reusable logic for common architecture services, thereby supporting "fix in one place." (2) GRNDS improves consistency across the application, as all application developers use it in a particular way. This consistency benefits maintenance. (3) GRNDS forces developers to follow strong architecture layering principles, dividing up where they put presentation logic, business logic, and data access logic. A well-layered system is more maintainable, as discussed in Section 3C.12.2 System Architecture Design.

Fundamentally, GRNDS reduces architecture costs and saves time. It gives the CCSAS project a head start, with a ready-to-use, robust toolkit.

3C.12.4.3 Proven Work Products

Software Architecture Design introduces the following work products:

- IEEE-based Software Design Description (SDD)

[22] "Open source" refers to a community of software developers and architects who share, maintain and enhance source code in an open but regulated exchange. It is "open" because developers have access to the source code, and because they are encouraged to fix bugs and suggest improvements. It is "regulated" because chief architects review proposed changes for quality and consistency. Accenture's community is "internal" because it is open only to Accenture personnel and the projects on which they work.

- Use Case Realization Diagrams
- Component Diagrams
- High-Fidelity User Interface Diagrams and Conversation Flows

The SDD is a composite deliverable that cross-references the other diagrams. The following paragraphs describe these products.

3C.12.4.3.1 IEEE-based Software Design Description (SDD)

The IEEE-based Software Design Description (IEEE J-STD-016-1995 Annex G.2.4) captures various architecture views of the software item. The SDD describes software item components (software units) that are implemented to fulfill the software requirements. The SDD serves as a communications medium between the software architecture design team and other parties interested in the software items. The SDD is analogous to the Rational Unified Process' Software Architecture Document.

While conceptually a single document, the SDD is delivered as a collection of references to actual design details. This allows for a more reasonable document where the details can be changed in only a single place. The following paragraphs outline the SDD sections.

3C.12.4.3.1.1 Software Item-wide Design Decisions

This section describes key features ofthe software item's behavioral design. That is, how it behaves from a user's point of view in meeting its requirements, ignoring the internal implementation.

3C.12.4.3.1.2 Software Item Architectural Design

This section:

- Identifies and describes the relationships between the software item components (software units)
- Describes how the software components dynamically interact
- Describes the design of the component interface

3C.12.4.3.1.3 Requirements Traceability

This section contains traceability from each software item component identified in the SDD to its originating software requirements. The manifestation of traceability between software design and software requirements resides in the design repository.

We document procedures for looking up the traceability links from SDD work products to their originating requirements. Where the tools support traceability, the procedures describe how to use the traceability features of the tool. For example, UML Use Case Realization Diagrams, which represent the design flow for a use case, are linked in

Rational Rose to the Use Case Diagrams, which themselves represent the requirements.

3C.12.4.3.2 Use Case Realization Diagrams

This work product relates the use case (and hence requirements) with the software implementation (and hence design). A realization diagram describes a pattern of interaction between actors and components, arranged in a sequential order. It shows the components participating in the interaction by their lifelines and the messages that they send to each other. A use case realization captures the behavior of a single use case. The realization diagram is linked to the use case for traceability.

Use case realization diagrams tie functional requirements to the design, thereby providing traceability.

Unlike class or component diagrams, which show static relationships, a use case realization shows dynamic interactions.

We create UML use case realization diagrams in our object modeling tool. Our diagrams go from the actor to the component (as opposed to class) level. As defined earlier in this document, a component is a portion of reusable software with a well-defined, public interface.[23] A component's internal implementation consists of multiple classes.[24] The mechanics of this internal implementation (i.e. the particular classes and their interactions) are a concern of detailed design. This is why *design* diagrams progress to the component level.

[23] Specifically, these are Java *interfaces.* In the J2EE specification, a Java session bean is a prime example of the type of components to which we draw Use Case Realization Diagrams. Once the flow reaches a session bean's interface, we are not concerned with the internals of how that component satisfies the interface. The internals are part of detailed design.

[24] Specifically, these are Java *classes.* In Java parlance, classes "implement" an interface.

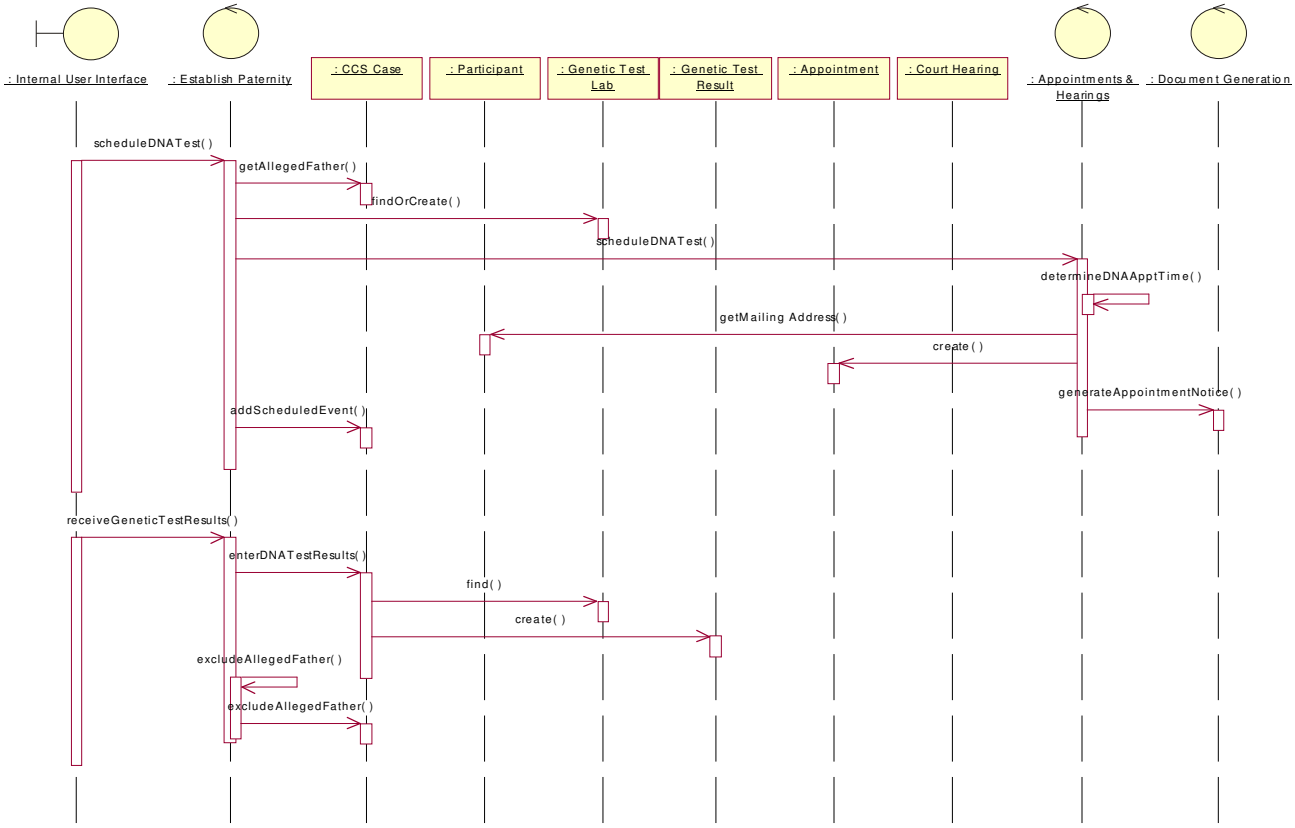


Figure 3C.12.4-15 Use Case Realization Diagram – *Realization diagrams are read from top to bottom. They clarify interfaces between components for complex flows.*

We draw components that participate in a use case realization as boxes at the top of the diagram. Actors typically appear on the top as well. The vertical line below each component, called the component's lifeline, represents the component's life during the interaction.

The Use Case Realization Diagram is a standard Rational Unified Process work product.

3C.12.4.3.3 Component Diagrams

Models help us understand a system by simplifying complexity and hiding details that are irrelevant to the task at hand. A component model is one of our most important work products. Section 4A.3.2 CSE System Architecture introduces the high-level business component model. Such a model provides a comprehensive but understandable abstraction of the software modules that make up the application. These encapsulated modules provide services through well-defined interfaces.

The Component Model illustrates the application components and their services.

The business component model is refined throughout design. It eventually becomes a UML Component Model, containing components, interfaces, and classes. During the coding phase, that component model eventually becomes Java classes.

We document the component model with UML, the Unified Modeling Language. This notation has been accepted in industry as the standard for modeling software. We believe it is important to use a standard, because developers are more likely to have experience with UML than with a proprietary notation. As well, UML is fairly robust, having improved through several revisions. Rational Rose, our object-modeling tool, supports UML.

With a tool, we model components graphically and document relationships between them. For each component, the model includes the component interfaces. Each interface documents the services, or public operations, provided by the component.

The Component Diagram is a standard Rational Unified Process work product.

3C.12.4.3.4 High-Fidelity User Interface Diagrams and Flows

"User interface diagrams" represent the users' views of the system. Users, which include State employees, NCPs, CPs, and employees of other agencies, have three types of views:

- Online interfaces, designed by the alliance with support from the State. We initially base these on the low fidelity user interface diagrams created during the Software Requirements Analysis phase. We formalize these hand drawings during Software Architecture Design.

- Online forms, designed by the State. Many forms must conform to Federal and State regulations. As such, they differ from online interfaces in that State personnel, as opposed to alliance personnel, create the forms' designs. Once the State has delivered the form designs to the alliance, the forms follow a code and test methodology similar to those of the online interfaces, mentioned above.

- Report interfaces, designed by the State. Similar to online forms, State personnel create the design of these forms to comply with Federal and State regulations. Given the report interface, the alliance determines how to map the data on the report interfaces to the reporting database. Taking advantage of reporting tools, it is not necessary to produce the same level of design documentation for development of reports. Once the mapping between interface and database has been designed, the alliance codes the report in a straightforward way, using the proposed Brio reporting tools.

These user interfaces, whether designed by alliance or State personnel, include:

- High Fidelity Page Layouts

- Conversation flows

The following paragraphs describe these work products.

3C.12.4.3.4.1 High Fidelity Page Layouts

High fidelity user interface diagrams illustrate the visual appearance of the user interface. This includes entry fields, list boxes, buttons, check boxes, radio buttons, field labels, menus, menu items, drop-down combo-boxes, as well as their placement relative to one another. These "hi-fi" diagrams are developed using a visual HTML editor or other graphics tool (like Visio or Powerpoint).

Page Layouts demonstrate how each page looks and behaves.

3C.12.4.3.4.2 Conversation Flows

Conversation flows provide a high-level overview of the pages and navigation routes for users. These diagrams are useful because web-based applications often contain many small, interconnected HTML pages. Conversation flows explain how users travel from one page to another, based on their actions.

Conversation flows show user interaction with the application at a high level.

This work product illustrates:

- HTML pages or forms making up a dialog
- Widgets and actions that transfer control from window to another (e.g., Clicking an OK Push Button may cause a results page to appear)
- Common pages that are available from most pages within the application, such as Help

We use conversation flows as input to user training documentation.

3C.12.4.4 Proven Process

Our robust design process for software architecture design has the following steps:

1. Identify Software Architecture Design Decision Makers
2. Adopt Software Architecture Design Principles
3. Create Use Case Realization Diagrams
4. Create Component Model Diagrams
5. Create High-Fidelity User Interfaces
6. Create IEEE-Based Software Design Description (SDD)
7. Review Architecture Design Along with Requirements
8. Obtain Acceptance for Design and Requirements

After following these steps for Version 1, we repeat them for Version 2. Version 2 software builds on Version 1 software and therefore builds on the Version 1 SDD. We expect Version 2 updates to primarily be extensions, as opposed to changes. (The former builds upon what is there, whereas the latter implies rework.) We handle updates in a controlled manner, using our processes described in Section 3C.4 Configuration Management.

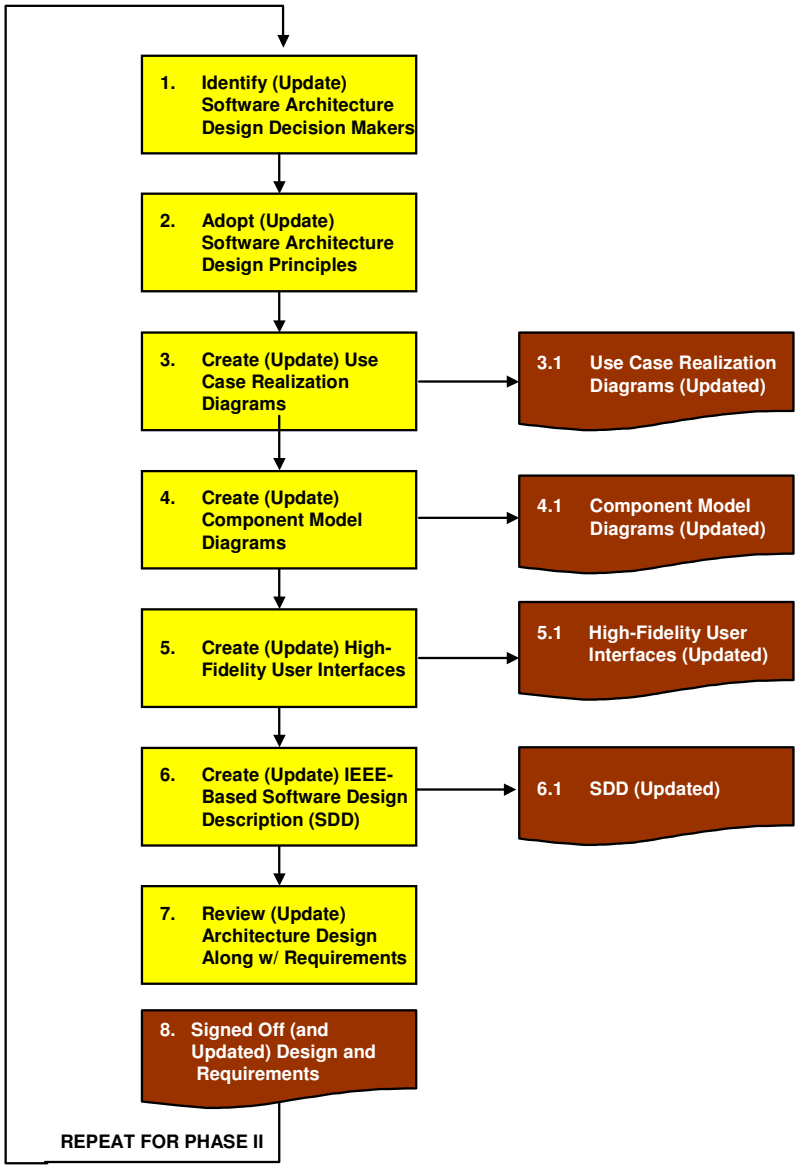


Figure 3C.12.4-16 Software Architecture Design Process – *Our SDDs include use case realization diagrams, component model diagrams, and high-fidelity user interfaces.*

The following paragraphs describe these steps.

3C.12.4.4.1 Identify Software Architecture Design Decision Makers

This step mirrors the step for identifying decision makers in Section 3C.12.2 System Architecture Design.

The Software Architecture Design decision makers are the individuals who review and concur with the submitted software architecture design.[25] The decision makers are most likely the same individuals participating in Software Requirements Analysis. We work with the State to identify these decision makers before beginning the design process.

Decision makers who were not involved in analysis should become familiar with the requirements before the design process begins.

3C.12.4.4.2 Adopt Software Architecture Design Principles

This step mirrors the step for adopting principles already documented in Section 3C.12.2 System Architecture Design. In addition to using the system design principles, we add the software design principles mentioned earlier.

3C.12.4.4.3 Create Use Case Realization Diagrams

Based on the use cases identified during the software requirements analysis, we create use case realization diagrams. Each diagram ties back to its original use case. We use Rational Rose to create these diagrams.

3C.12.4.4.4 Create Component Model Diagrams

The use case realization diagrams identify components (internal and external) that are required to fulfill the use cases of the software. The UML component diagram then describes the components in more detail, including their public interfaces and relationships.

This step produces a component diagram that shows each component that participates in the use case realization diagrams. It also shows their static relationships.

3C.12.4.4.5 Create High-Fidelity User Interfaces

In parallel with creating the use case realization and the component model, we create the high-fidelity user interfaces. This includes the conversation flows and the page layouts. For analysis we used pencil-and-paper diagrams. Now we document the page layouts in a software drawing tool or proper HTML editor.

[25] The authority of these “decision makers” does not replace the formal Contract Deliverable Acceptance Process (CDAP). Rather, the decision makers provide concurrence on the State’s behalf, before the deliverables enter CDAP.

We follow the usability standards defined as part of the System Architecture Design. We update these standards (under established configuration management processes) when appropriate.

3C.12.4.4.6 Create IEEE-Based Software Design Description (SDD)

After the components of the software have been identified, we create the IEEE Software Design Description (SDD). This is the key deliverable of Software Architecture Design.

Each functional software requirement maps to a use case (or multiple use cases) and its corresponding use case realization diagram. In turn, the use case realization diagram specifies which software components it requires. In this manner, requirements are accounted for and traceability between the SDD and the software requirements is clear.

As described earlier, the creation of this document should be treated as an overlapping process with Software Requirements Analysis. We use this document to help confirm and analyze the software requirements.

3C.12.4.4.7 Review Architecture Design Along with Requirements

This step mirrors the step for reviewing System Architecture Design with decision makers.

We submit the drafts of the SDD, Use Case Realization Diagrams, and Component Model Diagrams for review by the decision makers. As with the requirements review process, this initial review does not seek to obtain sign-off. Instead, its goal is raising questions and feasibility issues. We document issues at the initial session. But we resolve them in smaller break-out sessions with the appropriate parties.

3C.12.4.4.8 Acceptance for Design and Requirements

This step mirrors the step for System Architecture Design.

Break-out sessions resolve major issues. Then, the development team and decision makers formally review the Software Architecture Design deliverables, in light of their originating requirements. The goal of the software design is delivery of an SDD that is:

- Traceable to the requirements of the software item
- Externally consistent with the requirements of the software item
- Internally consistent between the components of the software item
- Appropriate considering the accepted design principles and standards
- Feasible for the software components fulfilling their allocated requirements
- Feasible for operations and maintenance

We then submit the Software Architecture Design for formal deliverable walkthrough. After we have identified and addressed major issues from this walkthrough, then:

- The design enters the formal CDL acceptance process.
- The accepted design is baselined into the configuration management (CM) process. We expect the design to evolve throughout the CSE development lifecycle. Proposed changes to both software requirements and architecture design are channeled through the process described in Section 3C.4 Configuration Management.

3C.12.4.5 Conclusion

We use proven architecture principles to create our software architecture design. These include layering and component-based design patterns for separating concern. We also build upon the SCP architecture objectives, best practices, and attributes, with which we agree.[26] Lastly, we leverage production-ready architecture assets with Accenture's GRNDS architecture.

We overlap the software requirements analysis and software architecture design phases. This validates and provides feedback on requirements. As well, we use pilots to determine feasibility (i.e., whether or not we can actually build to the requirements). This process improves the quality of – and confidence in – software requirements. When Software Architecture Design occurs in conjunction with Requirements Analysis, we engineer the system solution for success.

[26] See Section 4A.3.3 CSE System Architecture Objectives, Best Practices, and Attributes for our specific thoughts on individual best practices.

3C.12.5 Software Detailed Design

For detailed design, we build on our RUP work products with our own "Fast Track" detailed design approach. This innovative approach uses several new techniques to improve design quality and foster reuse. We bring more than an approach, however. We also leverage proven design principles, learned from years of experience designing child support systems and using component technologies. This helps us meet the State's requirements for a system that is maintainable, reusable, and scalable.

3C.12.5.1 Introduction

Out of the box, the Rational Unified Process (RUP) provides a very good start to detailed design activities. It contains roles, tasks, work products, and guidance on applying the methodology.

We believe in RUP, which is why we use it as a basis for detailed design. But we also have something to add. Over years of delivering child support systems and building with component technologies, the alliance has developed a "Fast Track" approach to detailed design.

This approach differs from traditional methodology steps for detailed design. Traditionally, a designer would have taken detailed design inputs, worked in a cubicle, created the detailed design, polished the documentation, desk checked it against standards, and then handed it off for thorough review. With the right review process, and with the right reviewers, this could ultimately result in quality. As well as a lot of rework.

We believe instead in moving the important team review to the *beginning* of the process. Below, we describe how we accomplish this, and how the "Fast Track" approach is different – and better. We also discuss some of the proven design principles that we use throughout detailed design.

We use a proven – and innovative – approach for building in quality, a proven approach for traceability, and proven design principles.

In addition to our innovative approach to detailed design, we continue with our robust approach to traceability. This involves using the right work products, the right mapping from work products in one phase to another, procedures for traceability, and a keen understanding of what tool support can provide.

3C.12.5.2 Proven Principles

The alliance uses proven principles for component design. Other software practitioners may have read about and may be able to discuss these concepts. But most lack the practical experience to apply the principles when it makes sense. Our principles include:

- 3C.12.5.2.1 - Size components appropriately
- 3C.12.5.2.2 - Avoid reinventing the wheel with design patterns
- 3C.12.5.2.3 - Use complex patterns sensibly
- 3C.12.5.2.4 - Reduce coupling

Many take a purist approach, applying concepts they read about in all situations, at all times. For example, in the name of flexibility, they create massive constructs and frameworks. This can backfire, resulting in an application that is overly complex and therefore difficult to maintain. Or one with too many different layers, with too much indirection, that performs poorly.

In contrast, the alliance uses proven practices and engagement experience to:

- Judge where different design principles are best used
- Weigh design tradeoffs
- Achieve a balance across non-functional requirements for maintainability, performance, and scalability

We know not only *what* principle to use, but also *when* to use it.

The following paragraphs describe our principles.

3C.12.5.2.1 Size Components Appropriately

We understand that there is an appropriate size for components. Large-grained components lead to:

- Greater encapsulation, because more details are hidden within a single component
- Loose coupling, because fewer components interact across component borders
- Greater manageability, as it is easier to configuration manage and deploy few things than many things

On the other hand, large components can get too large. When they contain too much functionality, they can be difficult to understand as a single unit. (At the extreme, consider an entire application built as one component.) They can be harder to maintain, because developers need to look through many more methods and many more lines of code to apply a fix. Moreover, large components reduce our ability to reuse only pieces that we want, without having to take the whole component off the shelf. Their size makes it harder to recombine them in new and unpredictable ways.

As we design the component model, we identify new components when we find sets of data and behavior that:

- Seem to go together
- Depend on one another
- Vary together

During re-factoring exercises, we may find components that contain small amounts of behavior. Or, some components may act as “garbage cans,” with miscellaneous methods grouped together without a higher purpose. In such cases, we merge and move their behavior into other components. When in doubt, early on we favor smaller grained components. Our experience has shown that it is typically easier to combine components later than to split them up.

Not too large, and not too small.

3C.12.5.2.2 Avoid Reinventing the Wheel with Design Patterns

Our developers’ guidebook includes proven, published coding patterns. A pattern is a named "nugget of insight" that conveys the essence of a proven solution to a recurring analysis, design, or coding problem. Design patterns provide solutions that are customized to solve general design problems in a particular context. These patterns may help to encapsulate application objects, decouple objects, increase application flexibility, centralize common code, or improve performance. For example, the *Singleton* pattern comprises code snippets for ensuring that only one instance of a class exists in a module at any one time.

Patterns are a formal way to document codified knowledge. They represent the thinking of highly experienced object technology practitioners.

Typically, these skilled resources rely on mental recall to apply rules-of-thumb as the opportunity arises. To capture this know-how, the software community has formalized many of these patterns. With recognized names like *Singleton,* they can be found in books, magazines, and web sites. This gives developers a shared language for reasoning intelligibly about recurring problems and proven solutions.

Furthermore, the maintenance effort benefits from a common language. Future CSE maintainers will likely have experience with many of the patterns used by alliance coders. Lastly, pattern formalization supports the knowledge transfer of expertise, an important concern to the CSE project.

Accenture brings to the alliance a multi-year investment that published an internal patterns handbook. These patterns suggest approaches for database access, state and memory management, screen validation, communication between the web server and the application server, as well as other topics. Accenture harvested the patterns across the globe, from large-scale projects like CCSAS. The alliance incorporates lessons learned from these patterns into the design of the CCSAS architecture.

3C.12.5.2.3 Use Complex Patterns Sensibly

It is important to use *complex* design patterns sensibly. These design patterns increase flexibility often at the cost of maintainability. A given pattern may provide the "ultimate" in flexibility. But that flexibility might require an algorithm that is overly clever or that prescribes excessive layers of indirection. Patterns need to be understandable by future CCSAS Project team members. They will gain such understanding by mentoring and on-the-job training.

Many patterns are not complex, and many patterns simplify CSE development. We do indeed use patterns, as mentioned above. But we use our experience to judge where they make the most sense, when the benefit justifies their use.

3C.12.5.2.4 Reduce Coupling

We reduce software coupling by locating functionality on components that use that functionality the most. When a component is intelligent enough to do most of its work independently, it avoids depending on other components. Loosely coupled components are desirable because they make individual components easier to change and maintain. Simply, a maintenance developer needs to worry about fewer interdependencies.

In contrast, a tightly coupled system degenerates into a "spaghetti" mess of interactions. This lack of modularity is brittle and hard to maintain. It may also degrade performance, if distributed components communicate excessively with each other. Additionally, the fewer dependencies there are between areas of the design, the easier it is to divide up the work between teams or individuals.

We balance the need to reduce coupling with the need to keep components from being bloated, or too intelligent, as mentioned above. We use detailed design work products to help detect excessive coupling. The UML component diagram shows static relationships and dependencies across classes. A component model with many links is one that cannot be easily divided into loosely coupled components. As well, the UML sequence diagram indicates interactions between components. We review these diagrams for chattiness between components. Such chattiness might suggest that the components should be merged or refactored.

3C.12.5.3 Proven Work Products

3C.12.5.3.1 Inputs

Key inputs into detailed design are:

- **High-Level Component Model** – represents the decomposition of system functionality into reusable software modules, or components
- **Use Case Realizations** – define the interactions between actors (such as end users or external systems) and CSE components
- **Software Requirements Specification** – captures the functional and technical requirements for the system software

- **High-Fidelity User Interface Prototype** – includes the look and feel of HTML pages as well as the conversation flow between them

Section 3C.12.4 Software Architecture Design has already described these inputs.

3C.12.5.3.2 Outputs

We document detailed designs by creating a detailed decomposition of software items identified in the System/Subsystem Design Description (SSDD) and further described in the Software Design Description (SDD). Our detailed design is documented using several methods, including:

- 3C.12.5.3.2.1 - Updated HTML Page Layouts
- 3C.12.5.3.2.2 - Updated UML Component Model
- 3C.12.5.3.2.3 - Java Class Model and JavaDoc
- 3C.12.5.3.2.4 - Class to database mapping

We configuration manage each work product with a unique identifier. The following sub-sections describe these work products in more detail.

3C.12.5.3.2.1 Updated HTML Page Layouts

We already created HTML Page Layouts in Software Architecture Design. See Section 3C.12.4 Software Architecture Design. These layouts provide a graphical representation of each HTML page, illustrating its “look and feel.”

During detailed design, the developer further refines the layouts. The developer writes comments that describe the page’s usage and behavior at a more detailed level. These comments describe each of the page’s widgets (e.g., button, menu item, entry field). This “Command-Action-Response” information describes:

- Control name and type (e.g., list box, radio button, entry field)
- The name of the activity that the page initiates as a result of the user action
- The visual response when a user interacts with a widget, such as selecting a menu item, clicking in a check box, or pushing a button
- Where the data came from, for widgets like entry fields and list boxes that display data
- Enablement and disablement of widgets (e.g., disabling a button when a user does not have access)

The page layout does not constitute a fully coded page. We wait to write its presentation logic until the coding phase. In detailed design, we simply embed additional design commentary in the HTML.

3C.12.5.3.2.2 Updated UML Component Model

We already created an initial, high-level component model in Software Architecture Design. See Section 3C.12.4 Software Architecture Design. This model provides a comprehensive but understandable abstraction of the software modules that make up the application.

During detailed design, the developer further refines the component model. The developer adds components, interfaces, and interface methods that were discovered in detailed design sessions. We update the UML component model using our object modeling tool.

During later coding, the component model becomes Java source code.

3C.12.5.3.2.3 Java Class Model and JavaDoc

3C.12.5.3.2.3.1 Detailed Designs in JavaDoc Format

Like a component, a class is an entity with a well-defined boundary and that encapsulates state and behavior. Class state is represented by attributes (the data owned by the class) and relationships; behavior is represented by methods (the behaviors the class can perform). While classes are fine-grained entities, components are typically larger. A component packages together one or more classes with related behavior.

UML class diagrams are one option for documenting a detailed design class model. These UML diagrams are good at specifying a class' name, methods, and fields. But they are less useful for storing formatted, easily readable business descriptions of the class because it may result in documentation that quickly becomes outdated.

Instead, the alliance makes use of *JavaDoc* for these textual descriptions. JavaDoc is a utility that extracts design comments from Java code.

During detailed design, the developer embeds specially formatted comments directly in the source code. These comments include HTML formatting instructions, recognized by the JavaDoc utility. Reviewers then use a simple JavaDoc command to extract those comments into a set of HTML pages. Naturally, this hyperlinked documentation is readable in a web browser. So reviewers can easily move from one class description to another or dive into more detail.

JavaDoc has the following advantages:

- It presents detailed design documentation in HTML. HTML's familiar navigation approach and structure is well understood by many. As such, JavaDoc does not require design reviewers to use a special tool or have specialized knowledge for how to use that tool.
- Detailed designers and less technical reviewers alike can discuss the same documentation.

We use JavaDoc for meaningful design commentary, stored directly in the code.

- Maintenance developers are much more likely to update detailed design documentation. This is because the higher-level, textual design information is embedded directly in the code, rather than in an external document.

Simply, our use of JavaDoc avoids shelf ware-bound documentation.

3C.12.5.3.2.3.2 JavaDoc Commenting Standards

Using JavaDoc properly requires more than just following special formatting instructions. It involves writing code comments with the appropriate audience in mind. Many developers are taught to write very technical code comments. For example:

```
/* Call data handler to access case file and get cases
/* for NCP; use where clause of Support? as yes.  Check
/* returned set length parameter for 0.  Populate list
/* by instantiating iterator over non-NULL hash map -- but
/* remember to call constructor with zero-OK flag and
/* catch collection_create exception, …
```

But our approach to self-documenting code makes additional, technical commentary like this is generally unnecessary. Moreover, these comments are cryptic – meaningful only to a technical audience. Instead, developers write comments that are more business-oriented. As in:

```
/* Do a quick existence check to determine whether the NCP is
/* attached to any cases with support-ordered amounts.
/* If not, don't show on list to avoid having the user
/* double-click and display a blank page.
```

3C.12.5.3.2.4 Class to Database Mapping

Another new work product in detailed design is the Class to Database Mapping.

While the CCSAS CSE application is object-oriented, its database is relational. Since we do not necessarily use a single object for each database entity, a mapping is necessary. This mapping identifies which objects can access which data entities, preserving object-oriented principles.

We perform this detailed mapping at the field level, specifying data types. We map from object attributes to columns on our Logical Data Model.[27] The mapping may result in a refinement of the class model, to include entities or attributes that were previously unaccounted for.

Design decisions that improve the database's logical consistency or performance are sometimes incompatible with good object-oriented design. For example, denormalization breaks up a maintainable,

The Class to Database Mapping links system and database development.

[27] And the columns in the Logical Data Model carry over into the Physical Data Model. See Section 3C.12.6 Database Design and Development, which describes these two models.

normalized data model for performance reasons. This further complicates mapping from a maintainable class model.

Section 3C.12.6 Database Design and Development discusses approaches for database access.

3C.12.5.4 Proven Process

The alliance's "Fast Track" detailed design approach complements RUP with additional steps and techniques. The following paragraphs explain:

- "Fast Track" Detailed Design Overview and Benefits
- Roles
- Process Steps
- Traceability

3C.12.5.4.1 "Fast Track" Detailed Design Overview and Benefits

Key features of the "Fast Track" approach, which are discussed in detail below, are:

- The developer, a functional architect, and a technical architect work *together* on the design. This work is done up-front rather than in a review session, at which point re-work is more costly. Moreover, including members of the functional and technical architecture teams improves quality from both perspectives, and results in a balanced design
- The developer is involved throughout, thereby reducing hand-offs and building ownership
- Auto-generated JavaDoc design commentary remains up to date

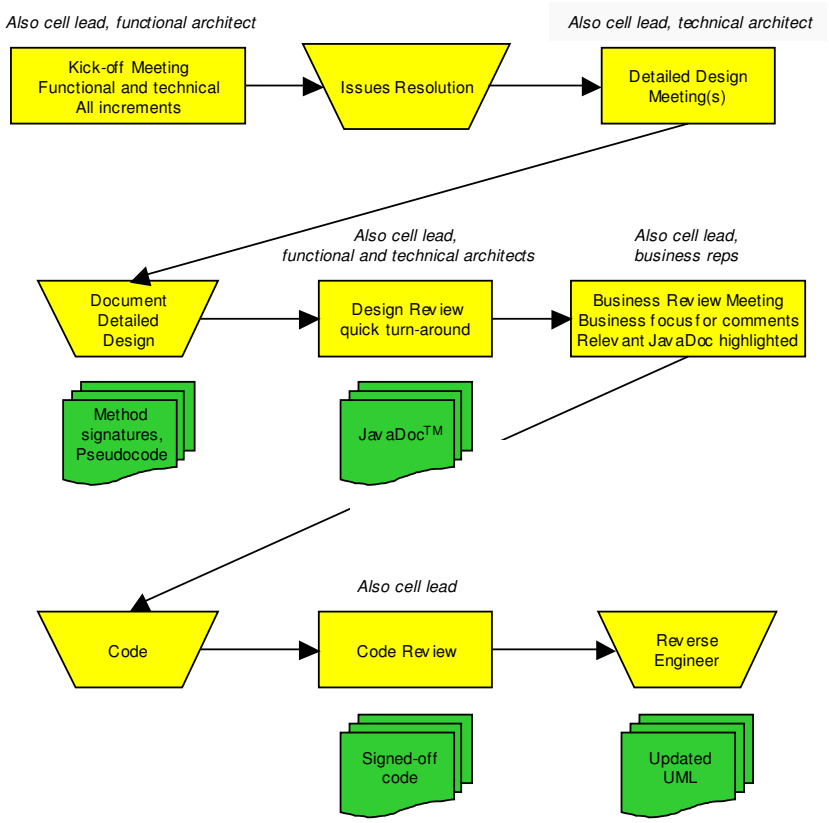


Figure 3C.12.5-1 "Fast Track" Detailed Design and Coding Process - *"Fast Track" builds in quality up front, keeps code traceable to designs, and reduces hand-offs.*

3C.12.5.4.2 Roles

The important players in detailed design are:

- The **developer,** who creates the detailed design work products and ultimately codes per the design. Developers report into the work cell lead, mentioned below.
- A **functional architect,** who is the champion for functional requirements and the creation of a reusable component model, across CCSAS CSE. Functional architects also report into the Application Development Manager, a role that is specified in the Project Management Plan (PMP) deliverable (PM 001).
- A **technical architect,** who understands ways to model components, apply design patterns, and use the technical architecture. Technical architects report into the Technical Manager (in the PMP).
- The developer's **work cell lead,** who is responsible for the overall quality of the design and its timely delivery. Work cell leads report into the Application Development Manager.

The detailed design process includes *both* technical and functional architects. Each of these individuals brings a different perspective and different information. For example, the technical architect understands well the proper use of technical frameworks or the guidelines for designing a particular type of Java class. The functional architect, on the other hand, understands the components that have already been designed and their relationships to the components under discussion. The functional architect is therefore in a good position to recommend opportunities for reuse.

3C.12.5.4.3 Process Steps

"Fast Track" includes the following steps:

- Kick-Off Meeting and Issues Resolution
- Detailed Design Meetings
- Documenting the Detailed Design
- Interim Design Reviews
- Code and Code Review

We repeat these steps for each set of detailed designs. Detailed designs are fairly self-contained, and we do not expect the internal designs of one component to significantly impact another component. However, in the event we do need to update work products, we use the processes described in Section 3C.4 Configuration Management. This might happen, for example, when a Version 2 detailed design results in updates to a Version 1 work product.

The following paragraphs explain these steps. Coding activities, in the last step, are also discussed in Section 3C.12.7 Software Coding.

3C.12.5.4.3.1 Kick-Off Meeting and Issues Resolution

The developer, functional architect, technical architect, and cell lead begin the detailed design process with a kick-off meeting. They review and confirm the design inputs. This includes the relevant Use Cases, the Software Requirements Specification, and the User Interface Prototype. During the meeting, the team clarifies the inputs and discusses open issues.

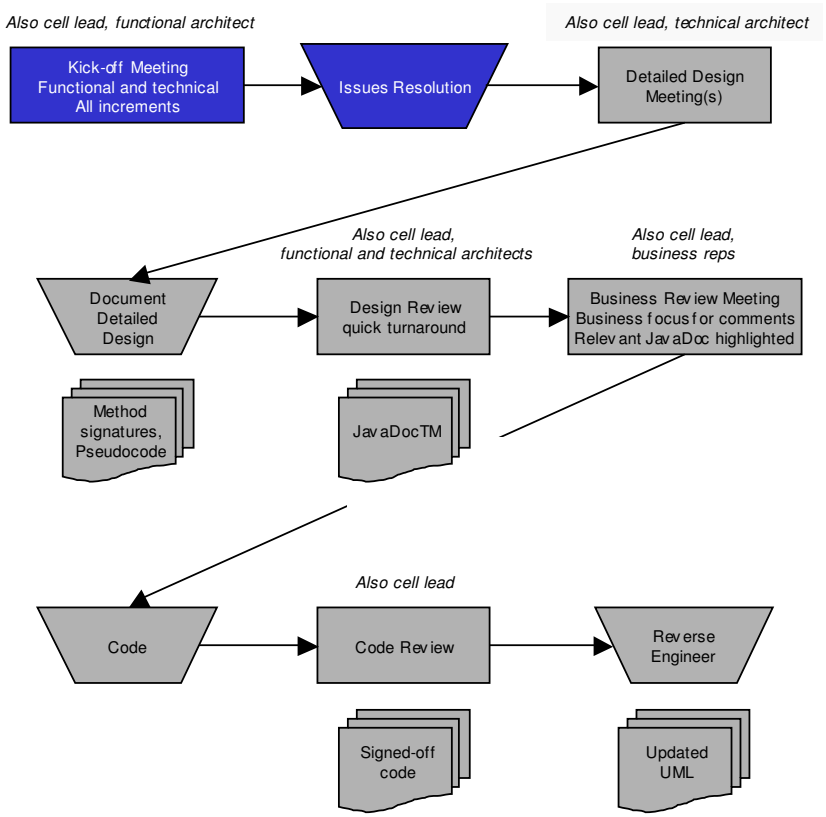


Figure 3C.12.5-2 Detailed Design Kick-Off - *This meeting involves the developer, functional architect, technical architect, and cell lead.*

Issues at this point are primarily functional. After the kick-off, the developer works with appropriate parties to resolve the issues.

Because of the developer's early involvement, the process avoids design hand offs. A hand off occurs when one individual performs design and then hands it off to a developer who is not familiar with it. In this way, the process also reduces design iterations and prevents rework, as all parties agree on the solution and tackle issues in the first design meeting.

3C.12.5.4.3.2 Detailed Design Meetings

With major issues resolved, the developer, technical architect, and cell lead host one or more detailed design meetings. This meeting is the cornerstone to the "Fast Track" approach.

The design group performs white-board design. Initially, they might choose to draw circle-and-stick diagrams to represent the interactions between different components. For each component, they agree on the interfaces and public methods. For each method, they fill out the method signature, including method inputs, input types, return type, and exceptions. In this manner, the architects and the developers reach consensus on the details of the design.

Design consensus comes at the beginning, rather than in a review session at the end.

The group focuses on encapsulation and other engineering principles that are architecturally significant. Internal implementation details are left to the developers to complete on their own. Such details include attributes, private methods, or the flow of code statements in a method.

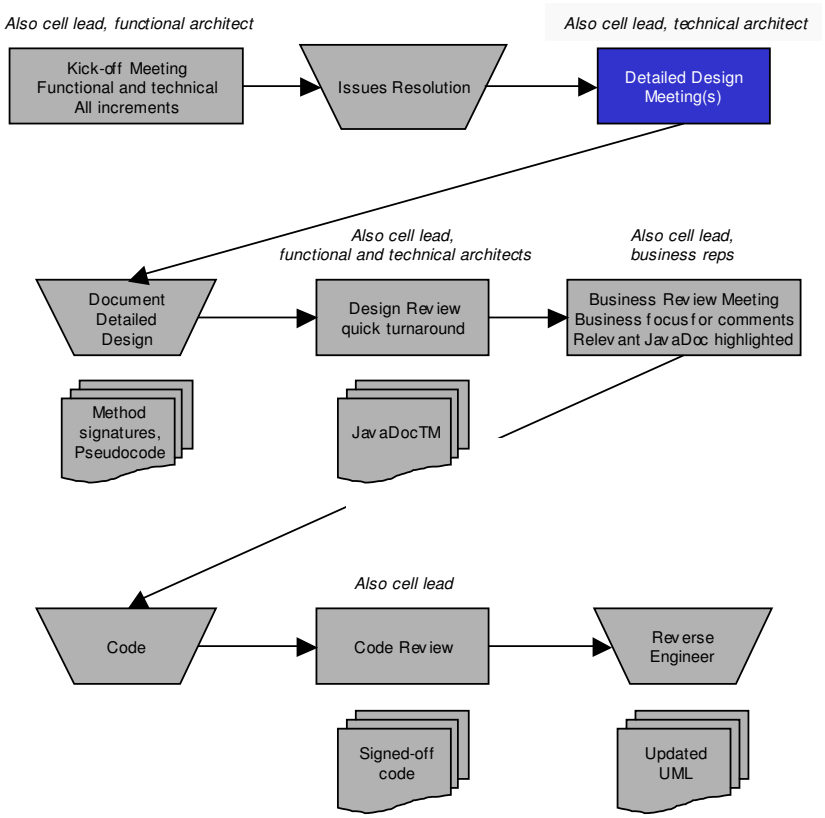


Figure 3C.12.5-3 Detailed Design Meetings - *Up front detailed design meetings are the cornerstone of our "Fast Track" approach.*

This approach simplifies the developer's job. Rather than starting with the design inputs and a blank design sheet, independent work begins with a roughed out, agreed-upon

design. The developer need only fill in the details. In this way, we build quality in up front. Later, we will inspect for quality during code reviews.

This is a key, differentiating point. If the team did not perform design in this manner, design reviews would likely uncover problems later. We would ask the hapless developer to re-factor the code, to reuse existing but forgotten classes, to better conform to the architecture, and to clean up public methods. Such changes often require much re-work.

Instead, our up-front design sessions focus on high-value areas such as the detailed class model, interfaces, and public methods. This addresses complex and high impact issues likely to affect the developer. We leave the less complex implementation details for the developer to solve independently. With preliminary design guidance, the developer is more likely to create a high quality design, and likely to create it more quickly.

3C.12.5.3.3.3 Documenting the Detailed Design

Developers are responsible for documenting the detailed design work products. They understand the design well at this point, because they participated in the up-front design meetings. This contrasts with approaches in which designers throw their designs “over the wall” to coders.

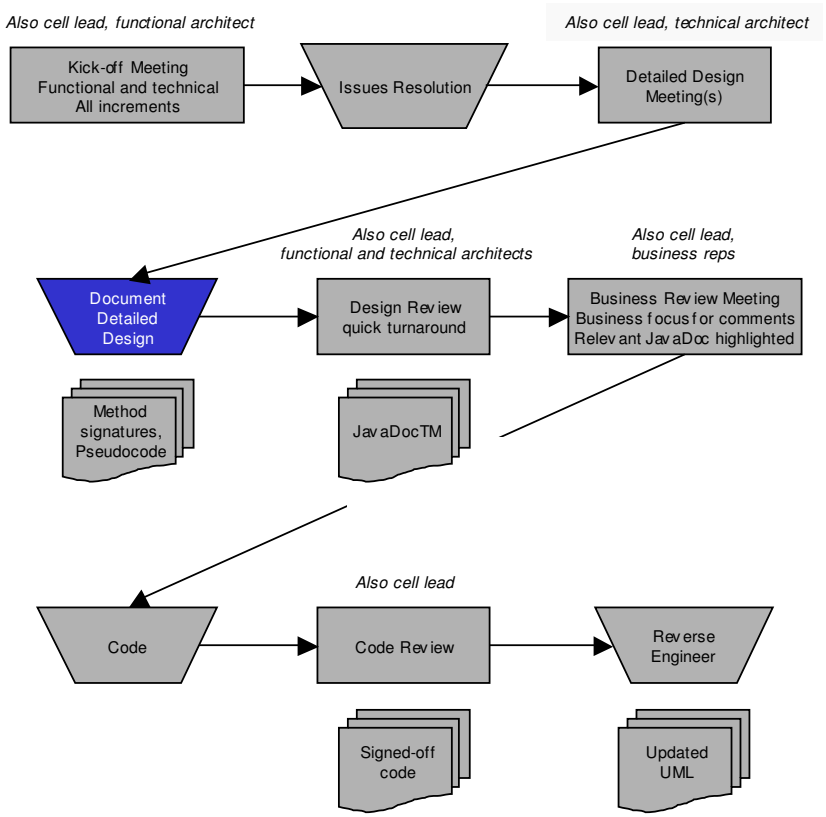


Figure 3C.12.5-5 Documenting Detailed Design - *We document the detailed design, primarily using UML and OO modeling tools.*

Following the detailed design meeting, the developers document what was agreed upon. They create or update the relevant work products comprising the detailed design, such as JavaDoc. See the subsection on "Proven Work Products" in this document.

3C.12.5.4.3.4 Interim Design Reviews

A design meeting covers a lot of ground. It touches the public methods, the method inputs and their types, exceptions, and important private algorithms.[28] Although the developer steadfastly takes notes, an opportunity for misunderstanding exists. Especially as the design group refines and re-factors the model, considering several different options for a particular class or method.

We reduce misunderstandings with an *interim* design review, following these steps:

- The developer documents the meeting results while they are still fresh in memory

[28] *Public methods* are those subroutines on a class that can be called by other classes. (As opposed to private methods, which are internal subroutines that no one outside the class can call.) *Method inputs* are the parameters passed along to a method when it is called. *Types* refer to additional programming information required for method inputs. Exceptions refer to programming errors that a method can raise.

- The developer captures these results in the Java class model and JavaDoc
- The developer submits them back to the group in a quick period, such as a few business days from the completion of the design sessions
- The technical architect, functional architect, and cell lead review the developer's documentation for accuracy. These three sign off on the detailed design package. This should be a straightforward review, with no surprises, as these same people participated in the design sessions.

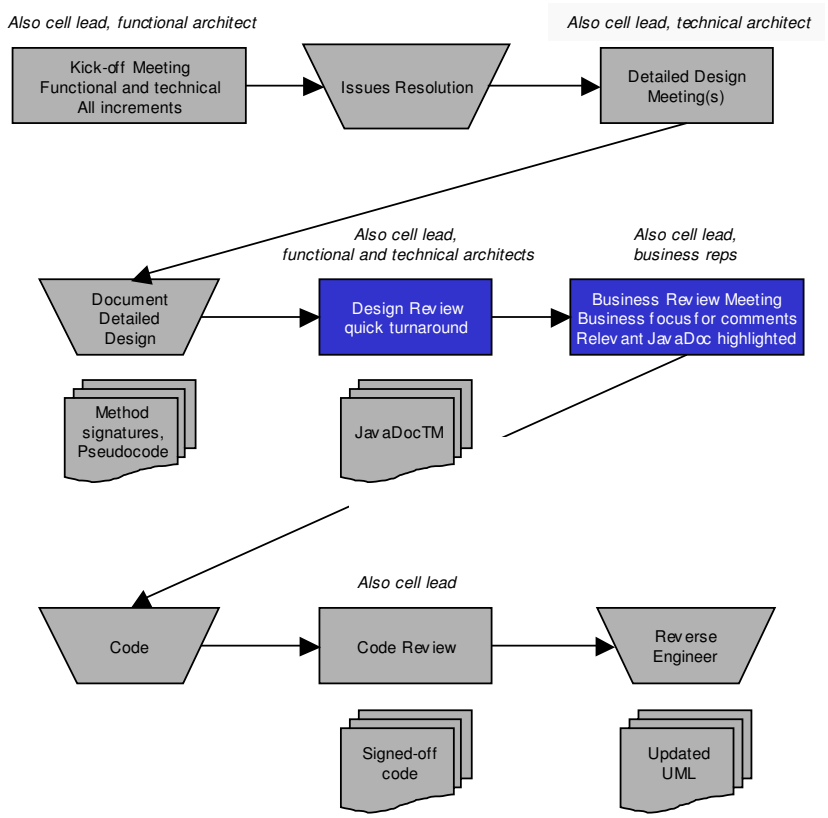


Figure 3C.12.5-5 Interim Design Reviews - *Reviews verify that the class model, public methods, and interactions are both functionally and technically sound.*

A second design review is then conducted with a business representative. This meeting reviews page layouts and corresponding "Command-Action-Response" information. C-A-R indicates how each widget responds to a user action. For example, what happens when a button is pushed or when the user exits a particular text field. This review also discusses the enablement settings for each widget. For example, when a button on the page should be enabled for one group of users but disabled for another.

The meeting also reviews design comments that outline the computational view of business rules and policies. We embed these design comments directly into the code

using JavaDoc. This improves the likelihood that design comments are maintained as code changes. Developers write JavaDoc in their Java classes. A utility extracts the JavaDoc in hyper-linked format, so it can be read more easily by less technical audiences, in a web browser. As JavaDoc is actually built into the code, it is discussed in detail in Section 3C.12.7 Software Coding.

The review with a business representative keeps us on track functionally.

Section 3C.8 Technical Reviews mentions these informal, detailed design reviews.

3C.12.5.4.3.5 Code and Code Review

Next, the developer builds out the implementation details of the Java class model. The class model includes the definition of Java classes, their operations, and their attributes. The developer fills in the bodies of the class and method shells, using the JavaDoc comments as a guide.

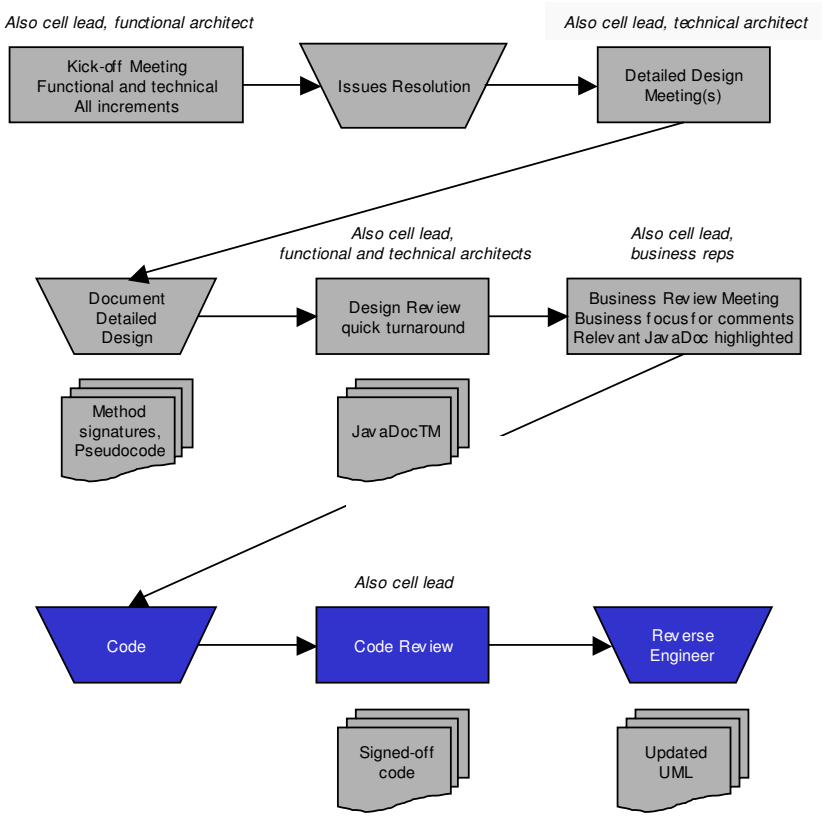


Figure 3C.12.5-6 Code and Code Review - *Fast Track continues from detailed design into coding.*

Section 3C.12.7 Software Coding discusses the coding process, proven techniques, standards, and reviews in more detail. The process step was included here to show its

place in the “Fast Track” approach. Section 3C.8 Technical Reviews mentions informal code reviews.

3C.12.5.4.4 Traceability

The internal implementation details of each CSE component are described with a class model. A feature of the “Fast Track” approach is that the detailed design class model actually becomes the coded class model. No synchronization, no round trip engineering is required to maintain any relationships between the detailed design of classes and their actual implementation. The JavaDoc provides us with a mechanism to extract the detailed design information in readable HTML. In this way, people who want to review the detailed design model – as separate from the code – can do so without being mired in the details of thousands of line of code. The JavaDoc provides an abstraction, or a high-level view, of the software code. But at the same time, the JavaDoc is embedded in the software code, so that we naturally maintain it.

The one-to-one relationship between JavaDoc and Java classes makes it straightforward to trace forwards or backwards from the detailed design to the code.

As well, our HTML Page Layouts (updated in detailed design) are directly traceable to our HTML Pages (code). This is because we draw our layouts directly in an HTML editor, rather than a separate design tool. In this way, our layouts become our actual, fully coded HTML pages.

We provide traceability for detailed design work products. Our traceability also works backwards, so we can evaluate changes to code as well as changes to requirements. We use tool support for this traceability, in addition to documented procedures for mapping work products from one phase to the next.

3C.12.5.5 Conclusion

The alliance’s “Fast Track” approach to detailed design not only *inspects*, but more importantly *expects* quality. This innovative approach:

- Builds quality in up front in detailed design meetings, with the right participants
- Keeps detailed designs and code synchronized through JavaDoc
- Reduces hand-offs by involving coders in the detailed design process
- Leverages experienced designers by inviting them to the design meetings
- Reduces errors and re-work by designing to the appropriate level
- Results in the proper use of both the functional and technical architecture

Finally, we leverage our significant experience with the detailed design of component-based systems. We bring proven detailed design principles. We guide designers to apply these principles at the right times, all the while weighing trade-offs to balance maintainability, performance, scalability, and cost of development.

Quality engineered throughout the process, not just at the end.

3C.12.6 Database Design and Development

A child support application is only as good as the data structure that supports it. The database design significantly affects an application's performance, reliability, and maintainability. We leverage proven design principles and techniques for this design. As well, we use robust approaches for managing design activities, from the early detailed planning to the generation of the last DDL script.

3C.12.6.1 Introduction

CCSAS' large scope and large scale require a well-thought out database design and implementation. HTML and Java design must be done in conjunction with the database design. For, applications need to be intelligent about their access paths into the database, rather than simply joining and selecting any data they need. Both application and database design depend intimately on each other.

Our team has decades of experience building reliable, scalable relational databases. We leverage proven practices learned on other RDBMS efforts of similar size and scope. Our approach includes:

- Conducting detailed, up-front planning
- Working with key functional and technical personnel to capture requirements in a traceable manner
- Designing and building the database according to established standards and proven practices

Our database techniques balance maintainability and performance. Our database development procedures focus on controlled, measurable progress.

3C.12.6.2 Proven Principles

Section 3C.12.2 System Architecture Design discusses the advantages of architecture layering. We use the data access layer introduced in that section to separate out business logic from database access. This improves maintainability and portability.

At a lower level of detail, Section 3C.12.4 Software Architecture Design gives an example of a design pattern to separate out data mappers from business objects.

In addition to these principles already described, database design introduces the following new principles:

- Grow data model over time
- Leverage the Data Architecture Framework
- Avoid naïve database access
- Use proven logical data modeling practices

- Use proven physical data modeling practices
- Model with an enterprise-wide data dictionary
- Use a frameworks-driven approach for database access modules

The following paragraphs describe these principles.

3C.12.6.2.1 Grow Data Model Over Time

We propose implementing CCSAS CSE in two versions. The first version delivers Statewide Services for locate, reporting, and the SDU. A project with a short-term focus might implement the data structures required to support this functionality without regard for CCSAS capabilities in future versions.

We offer a different approach. We believe it is important to have a holistic view of the data model before designing portions of that model. This results in a maintainable model and reduces the likelihood of rework in Version 2.

As such, we design the logical data model for both versions at the beginning of the project. With this big picture in mind, we design the physical data model for Version 1 functionality. The physical data model for Version 2 can come later with little expected impact on Version 1 data structures. Version 2 will build upon the database design of Version 1. In this manner, we reduce the number of throwaway data structures.

Moreover, we build the best of breed data model in Version 1. We do not simply copy data structures from the county systems (e.g. data columns or relationships particular to CASES or ARS). We may inherit particular data attributes that we use in Version 1 for temporary translation and historical purposes only. But overall, the Version 1 model is maintainable and extensible for the best of breed application.

We build the best of breed data model starting in Version 1.

Finally, the alliance also brings deep understanding of child support data structures from successful implementations in multiple states. This helps us kick-start the logical data modeling effort. We need not spend much time re-hashing the basics. Instead, we spend our time focusing on California's needs and on properly architecting the data.

3C.12.6.2.2 Leverage the Data Architecture Framework

Throughout planning and implementation, we leverage Accenture's Data Architecture and Data Management Analysis Framework. This promotes a cohesive understanding of data usage in all areas of the CSE enterprise. It also covers the logistics of how data is physically stored, moved, and accessed.

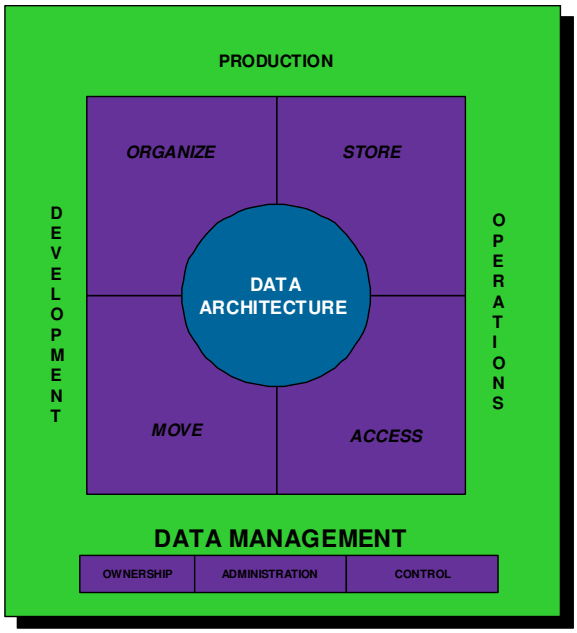


Figure 3C.12.6–1 Database and Database Management Framework - *This framework simplifies how we think about data architecture.*

The framework addresses the following categories:

Organize	*Store*	*Move*	*Access*
• Logical Data Modeling • Data Distribution • Data warehouse architecture design • Metadata	• Data quality • Physical database design • RDBMS and VLDB concepts and architectures • Data sourcing	• Data movement and replication tools/techniques; • Extract/Transform/Load (ETL) tools/techniques; • Application and data integration tools/techniques; • Data flows (within and between data stores); and • Audit and reconciliation	• Application and query design for efficient data access (data access layer) • Reporting architectures, tools and techniques • Technical architecture design for scalability, availability and performance • Stress and performance testing

3C.12.6.2.3 Avoid Naïve Database Access

Equally important to the database design is how the application accesses the data. The most highly tuned database design is meaningless if application developers join across too many tables, or from the application make too many trips to the database to retrieve data.

Naïve application of object-oriented techniques to data access can cause significant performance problems. We avoid purist techniques, which are often promulgated by object-oriented experts – experts who have yet to deliver a large-scale, mission-critical system.

Object-based retrieval is such a technique, that needs careful attention. With object-based retrieval, each business class has its own access module. This access module is reusable: the Java class can be used in several different transactions but with the same data access code. Developers can grab the access module "off the shelf" without coding any new SQL statements or stored procedures.

This is compelling. But, as described, there are performance implications.[29] Consider a particular transaction that updates a dozen business objects. This requires a dozen trips, across the network, and a dozen separate database I/Os.

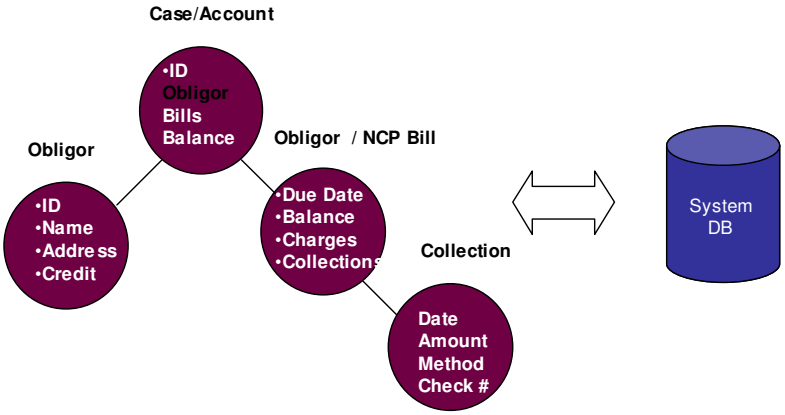


Figure 3C.12.6-2 Object-based Retrieval - *Object-based retrieval is reusable. But, unfortunately, it degrades performance in large systems.*

One issue can be that object-based retrieval can typically returns more data than is required for a particular window. This is because a data access module always returns the same data attributes for a particular business class, regardless of the calling window. Granted, this may have little noticeable impact on a window with a few objects, but it can be inefficient for a window with a scrolling list box that only needs to display a few summary fields for each row.

Most traditional client/server systems skipped object-based retrieval by hand-crafting data structures specific to each transaction. This resulted in one-off data structures that

[29] One can improve on the naïve approach by wrapping the reusable, CRUD access modules with transaction-specific modules. That avoids some of the network performance problems mentioned in the main text, although the multiple I/O calls remain.

were not reusable. An ”impedance mismatch” therefore exists between reusable, whole objects and specific transactions that need only parts of those objects.

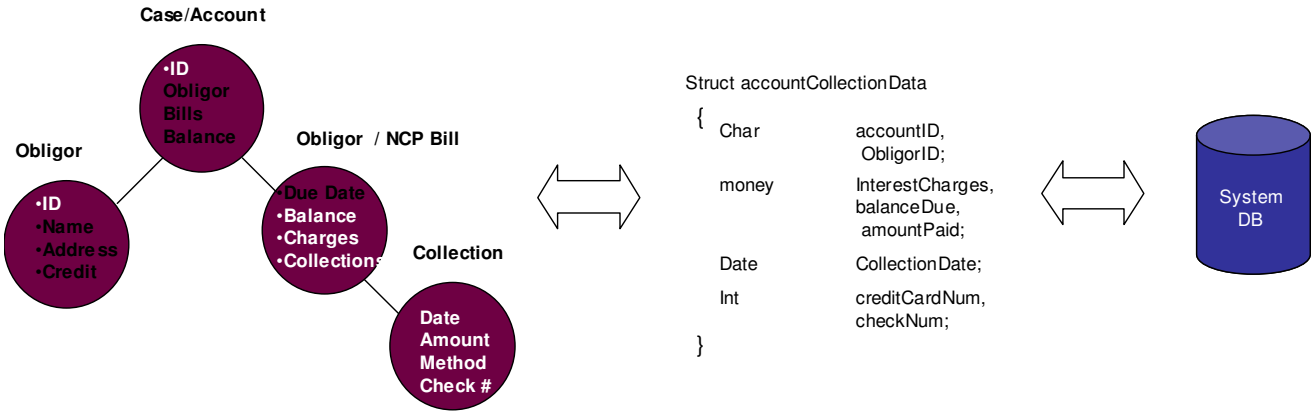


Figure 3C.12.6-3 Getting the benefits of performance as well as the benefits of reuse - *Transaction-dependent retrieval for performance meets transaction-independent objects for reuse.*

One way this we solve this mismatch is by using a proven, balanced approach:

- Transactions are specific to a window, highly tuned, returning only the necessary data. This improves performance; but at the same time
- We avoid creating class definitions that are particular to a specific transaction. Instead, our business class definitions remain reusable across transactions. The key is that we *partially* fill instances of the class with the data that was brought back. In this manner, we bring back only the data required of a particular transaction, but multiple transactions use the same business class definitions. This improves reuse.

For over a decade, the alliance has been building large-scale, object-oriented systems that map to relational databases. We understand the pitfalls and the trade-offs; we have proven solutions.

3C.12.6.2.4 Use Proven Logical Data Modeling Practices

The following paragraphs discuss proven techniques learned on other projects with similar functionality or similar technologies. These include:

- Combine top-down and bottom-up analysis
- Involve users and object modelers
- Normalize the LDM to the third normal form

The following paragraphs describe these practices.

3C.12.6.2.4.1 Combine Top-Down and Bottom-Up Analysis

Our experience has shown that a combination of a top-down and bottom-up analysis approach works well. The top-down approach helps maintain consistency and is particularly useful during initial modeling activities. Starting with the top-down view means starting with the big picture. This is more likely to result in a well-factored, normalized, consistent LDM. Such an LDM would not grow organically from independently designed tables any more than a reusable component model would easily emerge from independently designed components.

We complement top-down analysis with a bottom-up approach that helps complete the details. Bottom-up analysis (a.k.a. canonical synthesis) may reveal non-obvious attributes that are nevertheless required by the business. Our bottom-up analysis might include searching through the attributes of existing screens, reports, and the file layouts obtained during data conversion mapping activities.

3C.12.6.2.4.2 Involve Users and Object Modelers

We involve users in initial design discussions. Our experience has shown that users often have an accurate view of the "real-world" complexities associated with the data. For CSE users, we include the decision makers already identified in Sections 3C.12.2 System Architecture Design and 3C.12.4 Software Architecture Design. This has the benefits that:

- They are already familiar with the requirements and overall design approach
- They have a relationship with the other team members
- Keeping the same people involved in system design, software design, and database design improves integration between the three

Additionally, we include alliance object modelers in database design discussions. This helps marry the application and database structure. As well, object modelers can help conserve object-oriented concepts, like inheritance, for the LDM.

3C.12.6.2.4.3 Normalize the LDM to the Third Normal Form

Normalization is the process of organizing entities and attributes in a relational data model to remove ambiguity and redundancy. The stages of normalization are referred to as normal forms. They progress from the least restrictive (First Normal Form) through the most restrictive (Fifth Normal Form). We normalize our LDM typically to the Third Normal Form. There are many reasons that Third Normal Form has become a useful level for logical modeling:

- After Third Normal Form, normalization becomes far more theoretical and much less practical
- Third Normal Form is sufficient because it eliminates all non-key attributes that do not depend on the key

- Dr. E.F. Codd, the father of relational database design, originally defined three levels of normalization

3C.12.6.2.5 Use Proven Physical Data Modeling Practices

The following paragraphs discuss proven techniques learned on other similar projects. We:

- Balance current requirements and future proofing
- Review with appropriate parties
- Avoid meaningful keys
- Denormalize judiciously
- Partition vertically and horizontally
- Focus capacity efforts on high-impact areas

The latter three relate specifically to performance, a concern we address during physical data modeling.

3C.12.6.2.5.1 Balance Current Requirements and Future Proofing

Our experience shows that design should meet predicted requirements five years away. Excessive future-proofing is expensive and may be moot. On the other hand, building only for current requirements is shortsighted. We believe that five years is roughly the right compromise.

3C.12.6.2.5.2 Review with Appropriate Parties

We have found that the database model should be jointly reviewed with the database administrator and application design team members. Program designs and code depend intimately on the database design. To avoid rework, application designers must understand and agree with this design.

3C.12.6.2.5.6 Avoid "Meaningful" Keys

We avoid "meaning" in primary and foreign keys. For example, some systems use social security numbers – a meaningful attribute – to identify people. This is fraught with problems: social security numbers change and people sometimes share the same number for employment.[30]

Proven practices dictate that keys be unique, meaningless, and immutable. So we instead use algorithms to generate a unique key, rather than using existing columns to create a composite key. Any change to a meaningful primary key is likely to have extensive consequences, since every reference as a foreign key must be modified. Any attempt to build meaning into a key is risky, because the key may one day need to change for functional reasons.

[30] The reader should not think that "meaningless" here means not important. Rather, the point is that the keys are functionally meaningless. (This is an industry standard term.)

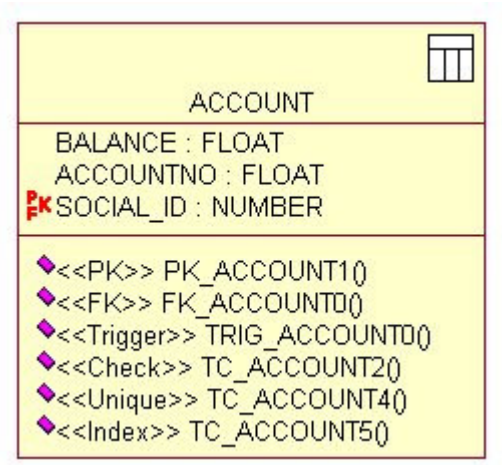


Figure 3C.12.6-4 Avoiding Meaningful Keys - *This meaningful key is composed of Social Security Number, which is neither unique (due to fraud) nor immutable (due a person's SSN changing).*

3C.12.6.2.5.3 Denormalize Judiciously

Normalization is the process of organizing entities and attributes in a relational data model to remove ambiguity and redundancy. Denormalization, on the other hand, is the process of changing the design by deviating from the rules of normalization to meet performance objectives. For example, denormalization techniques to avoid table joins include adding redundant data to a model, creating summary data, or combining multiple tables.

Application performance and the need for denormalization are ongoing considerations during physical database design. Obvious candidates for denormalization may be identified based on the application flow, high-level program definitions, transaction volumes, and the results of affinity analysis. Throughout the process, rules of thumb may be used to identify potential performance problems (e.g., update programs should not perform more than thirty database I/Os).

During physical database design, call patterns and the results of performance modeling and benchmark testing may reveal the need for further denormalization. Moreover, subsequent performance testing results may also lead to more denormalization.

Regardless of the catalyst for denormalization or the denormalization technique used, we denormalize as needed to achieve performance targets but no more. While denormalization improves performance, it adversely impacts maintainability, one of the key CCSAS architecture objectives. We denormalize judiciously, then, by *starting* with a normalized model and then denormalizing particular tables only when:

- The call patterns, transaction volumes, or performance model suggest it would be beneficial
- We observe a performance problem through testing or other means

In other words, we focus our denormalization efforts…rather than hoping that ad hoc changes will improve performance.

3C.12.6.2.5.4 Partition Vertically and Horizontally

As with denormalization, we consider vertical and horizontal partitioning to improve performance. Vertical partitioning could involve dividing a very large table, with many rows, into multiple smaller tables, each with the same columns but containing different rows. Dividing a table by primary key supports parallel access and can speed queries.

Horizontal partitioning could involve separating a large, sparse, nonvolatile field (e.g. a textual "comments" field) into a separate table.

3C.12.6.2.5.5 Focus Capacity Efforts on High-Impact Areas

When estimating data volumes, we perform a sensitivity analysis to concentrate detailed analysis on those tables and assumptions whose marginal effect is highest. This avoids wasting effort on confirming assumptions with insignificant impact.

3C.12.6.2.6 Model with an Enterprise-Wide Data Dictionary

We use an enterprise-wide data dictionary, based on industry standards. The following paragraphs describe the approach and standards.

3C.12.6.2.6.1 Value of an Enterprise-wide Data Model

Enterprise data can be stored in a variety of sources, such as:

- Relational databases
- XML files, perhaps for B2B interchange with other agencies
- Data warehouses
- OLAP decision support tools

These data stores, and their associated modeling tools, each use their own definition and format for metadata. This makes the creation and sharing of enterprise-wide data, across storage technologies, complicated and error-prone.

Large projects like CCSAS, with multiple data sources, benefit from a shared, enterprise-wide, metadata model. This model is essentially a common data dictionary. For a particular attribute, the dictionary specifies type, definition, constraints, and relationships with other attributes. Then, for example, all CCSAS applications can share a common view of the Case entity, as well as its relationship to the Non-Custodial Parent entity.

A data dictionary provides consistency for data elements, relationships, and constraints.

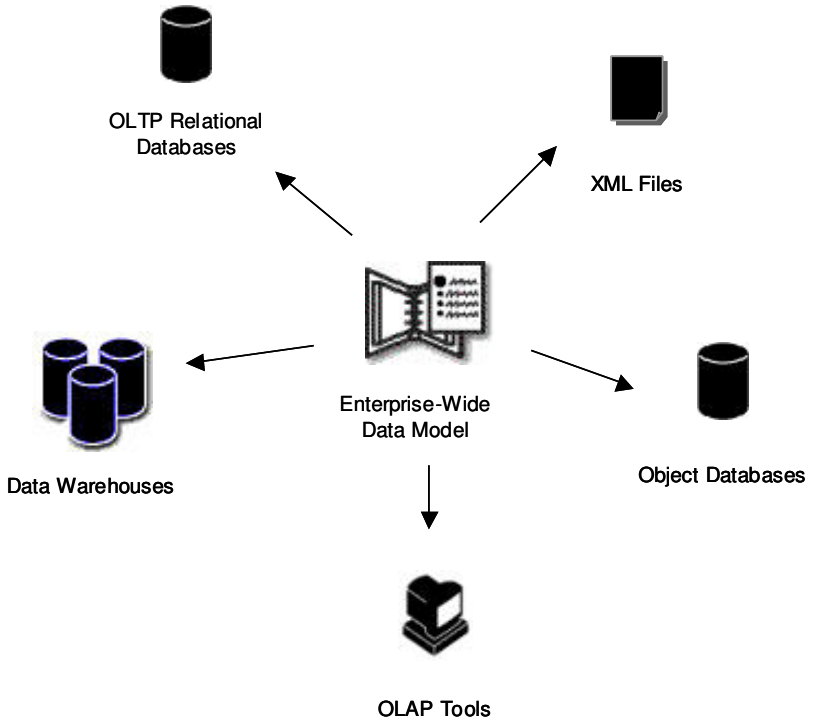


Figure 3C.12.6-5 Enterprise-wide Data Dictionary – *A common data model can provide consistency across several different formats.*[31]

Having an enterprise-wide dictionary then facilitates development of the CCSAS Statewide Services functionality.[32] The common definition of Case can be used as input to:

- A relational database model for storing Case information in the statewide database (which can then be generated into actual DDL)
- An XML file for gathering Case information from the counties
- The enterprise-wide component model including the Case class (which can then be generated into actual Java code)

We use the dictionary for building CSE modules, application-to-application (A2A) integrations, and business-to-business (B2B) integrations.

[31] Some of the formats, like object databases, are not within the scope of the project. The point, however, is that a data dictionary could be used for those types of formats in the future.
[32] Section 4.1 Business Solution Narrative describes the Statewide Services functionality.

3C.12.6.2.6.2 CWM Standard for Representing that Model

Storing and sharing enterprise-wide metadata requires:

- A **formal, well-understood language** for defining the structure and semantics of metadata
- **Tools** for modeling and storing the metadata
- An **access mechanism** for sharing metadata across tools and applications. This should include well-defined APIs and other techniques for getting information from the metadata repository

MOF is a standard for creating modeling languages.

The Object Management Group (OMG) has defined a well-recognized standard for modeling any type of metadata. It is called the Metadata Object Facility (MOF). MOF is flexible, and it can be used to describe modeling languages for:

- Designing objects (e.g., UML)
- Designing data stored in relational databases (e.g., ER diagrams)
- Designing document templates (e.g., Microsoft Word's style language)

CWM is a modeling language, based on MOF, for enterprise data.

Recently, the OMG defined a standard for representing a particular type of metadata: metadata for creating an enterprise-wide data model. This standard is called the Common Warehouse Metamodel (CWM). CWM defines a model within the MOF structure, or standard. The primary objective of the CWM is to define a metamodel[33] of a generic data warehouse architecture. It is a specification for modeling metadata for relational, non-relational, multidimensional systems, and most other elements of a data warehousing environment.

[33] That is, a model for storing metadata

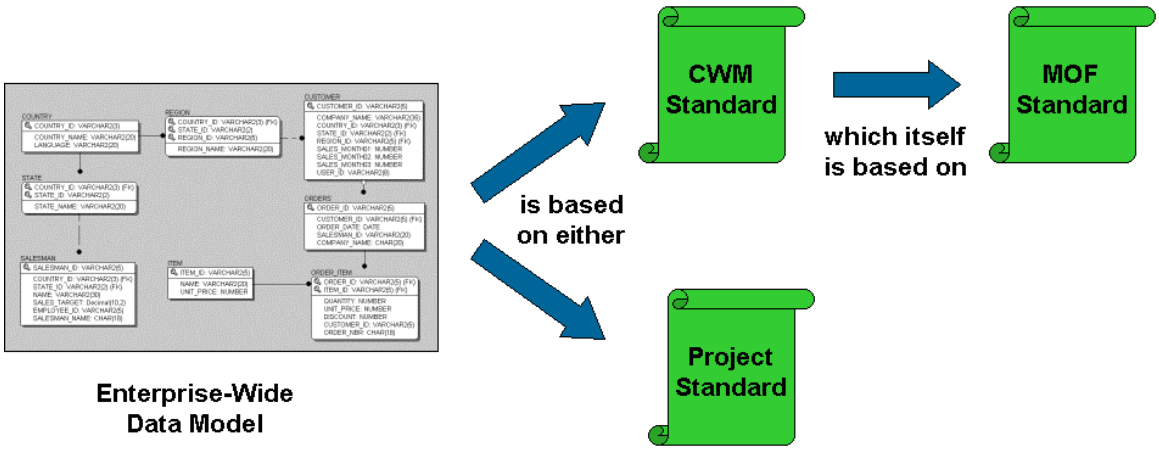


Figure 3C.12.6-6 CWM and MOF - *We base our enterprise-wide data model either on industry metamodeling standards or on well-defined project standards.*

CWM is the leading standard in this area, although it is not yet well established in industry.[34] CWM provides a formal, standardized language for modeling enterprise-wide data. As well, CWM standardizes on APIs for access. UML is used as the diagramming notation for representing CWM models. A UML-compliant tool can therefore be used to create an enterprise-wide data dictionary based on CWM. In this manner, CWM addresses the three points mentioned earlier in this subsection.

We consider applying CWM as we develop our project standards for the CSE enterprise-wide data dictionary.

CWM is comprehensive. It includes modules, or "packages," which are not required for CSE's needs. As but one example, CWM includes a package for multi-dimensional data modeling, which is not part of CCSAS requirements. We therefore only consider relevant packages of CWM for creating our enterprise-wide dictionary.

As the standard is relatively new, the number of products with official support for CWM is small. Robust CWM tool support may not be available in time for project use. In such a case, our contingency plan is to continue building an enterprise-wide dictionary and to develop a project-specific standard. This positions CCSAS CSE well for when the CWM tools do become available.

Metadata and metadata models can be a difficult concept to understand. In summary, to simplify:

- CWM is a standard for defining enterprise-wide data

[34] The MetaData Coalition (MDC), a separate organization from the OMG, had a competing but similar standard with its MetaData Interchange Specification (MDIS). Previously MDIS competed with CWM. However, to avoid fracturing the marketplace, the MDC recently decided to merge its work into CWM and make CWM the predominant, single standard.

- We consider CWM as we create the project standard for our enterprise-wide data dictionary
- Applying a standard facilitates using this dictionary for a number of data sources – be they relational databases, XML files, or otherwise

IBM brings to the alliance its active involvement in defining the CWM standard.[35]

3C.12.6.2.6.3 Use a Frameworks-Driven Approach for Database Access Modules

In a "frameworks-driven approach," the architecture drives how applications access the database. Individual data access or manipulation modules share many of the same reusable, technical modules. (E.g., snippets of code for getting a database connection and then closing it.) When using a standardized framework, design documentation need only focus on the variability across different software units. That is, what makes one software unit for data access different from another.

Experience has shown that a well-architected data access layer can reduce the variability between data access modules to simply:

- The name that uniquely identifies the data access module, so it can be called from software code (i.e., Java)
- The SQL statements it embodies for data access and manipulation
- Required inputs, in the SQL host variables for the SQL statement (e.g., parameters in a "where…" clause)
- Validation on those inputs
- Outputs, in the SQL host variables for the SQL statement (e.g., parameters in an "into…" clause)
- Error conditions

The technical framework handles the rest. In this way, designers need not worry about, nor repeatedly document, underlying technical details. Instead, they focus on the variability across data access modules. That is, they simply focus on "filling in the blanks" by specifying the access module name, SQL statements, inputs, outputs, and error handling to the framework.

3C.12.6.3 Proven Work Products

Detailed planning is the foundation for a high quality design and successful implementation. Our planning work products include the:

1. Database development plan

[35] We also note IBM's visible commitment to this standard in its product line: IBM's DB2 Warehouse Manager supports CWM for metadata interchange.

2. Database administration procedures (contained in the System Administration Plan CDL TM 025)

We leverage the lessons learned from the Data Architecture and Data Management Framework, introduced above, to create our plans. Following the plans, we create the work products for our database design, including:

3. Logical Data Model
4. Physical Data Model
5. Database Design Description (contained in the Software Design Description – CDL TM 021)

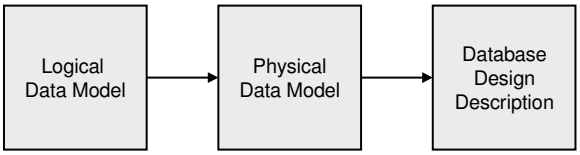


Figure 3C.12.6–7 Database Design Work Products - *Relationship of the key database design work products.*

Lastly, for the operations architecture we deliver:

6. Database Activity Report

The following paragraphs discuss these work products.

3C.12.6.3.1 Database Development Plan

We produce and execute a Database Development Plan that contains:

- Approaches for general database development activities, including development methods, standards, and traceability
- Approaches for detailed database development activities, including work planning approach, environments, configuration management, corrective action, technical and management reviews, and risk management
- Project organization and resources

We apply the same level of rigor and discipline to our database requirements, design, and implementation activities as we do to other software efforts.

Lastly, we update this plan periodically.

3C.12.6.3.2 Database Administration Procedures

We produce a Database Administration procedures that address operations management activities, including:

- Installing and configuring the DBMS
- Managing DBMS security
- Managing data storage
- Backing up databases
- Restoring databases
- Automating administrative tasks
- Transferring data
- Formatting and maintaining the DBMS
- Managing replication
- Providing high availability

We update these procedures periodically.

3C.12.6.3.3 Logical Data Model (LDM)

The Logical Data Model (LDM) serves as a design of the application's information requirements and relationships, without regard to their detailed, physical implementation. The LDM consists of diagrams showing the overall logical structure of the databases including entities, attributes, and relationships.

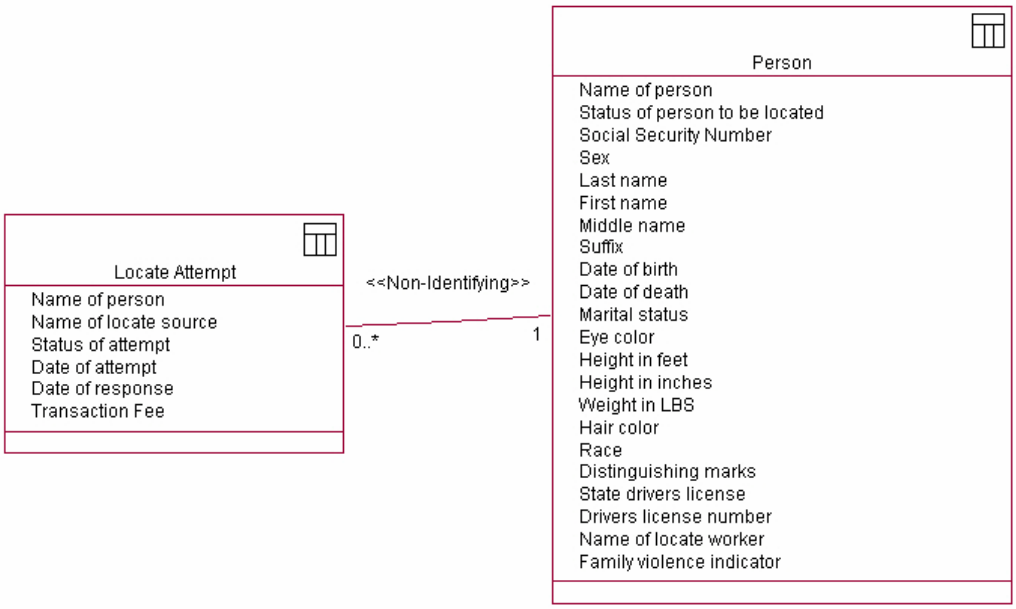


Figure 3C.12.6-8 Logical Data Model - *The logical model includes entities, columns, and relationships.*

3C.12.6.3.4 Physical Data Model (PDM)

We create and maintain a Physical Data Model (PDM), based on the LDM as a starting point. The PDM includes additional implementation information, like uniqueness keys, detailed type information, logging columns, constraints, or changes to the LDM for performance purposes.

We create the PDM in our data-modeling tool, using a well-known notation. We also use this tool to generate the database DDL and other related scripts, from the PDM. The "Proven Tools" subsection in this document discusses tool usage.

We consider detailed, technical aspects of database development for the PDM. This might include:

- Buffer pools
- Tablespace partitioning
- Space allocations
- Locking level
- Index uniqueness and clustering
- Index sequencing (ascending or descending)
- Physical storage device assignments

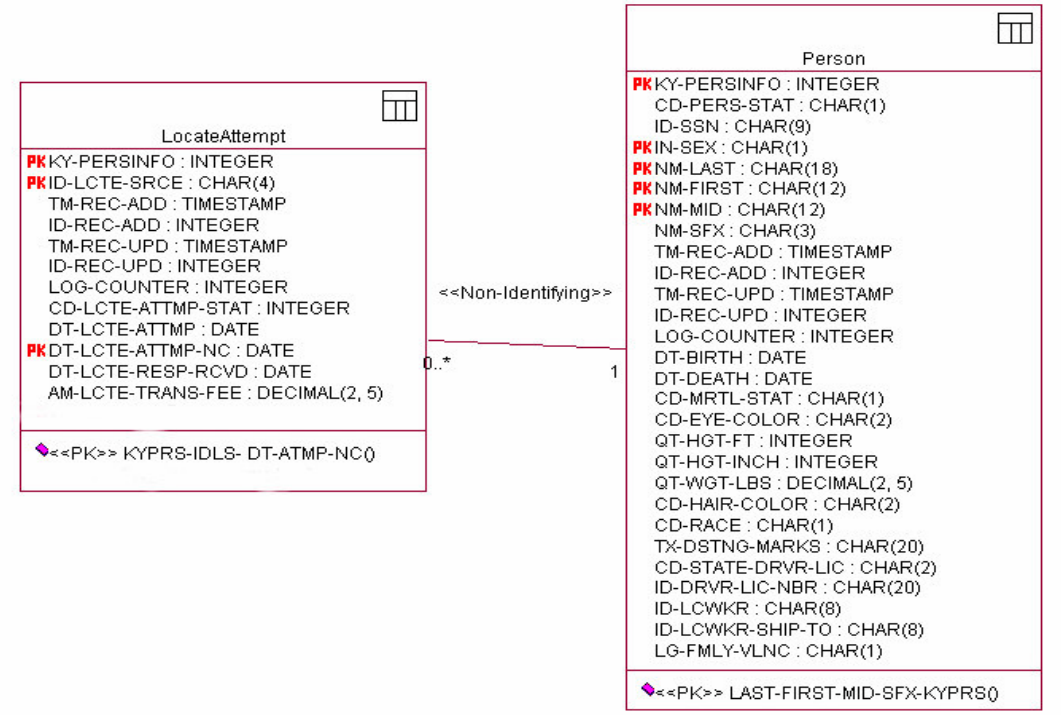


Figure 3C.12.6-8 Physical Data Model - *The physical model includes additional implementation information that is not found in the logical model. And note the use of a "meaningless" primary key on Person, as recommended elsewhere in this document.*

3C.12.6.3.5 Database Design Description

We create a Database Design Description as part of the Software Design Description. This product outlines system wide database design decisions. For example, this might include the approach to referential integrity, backup/restores, or chosen levels of normalization. The DBDD also cross-references the physical and logical data models. These models are configuration managed and stored in a database modeling tool.

3C.12.6.3.6 Database Activity Report

We deliver an ongoing Database Activity Report for the operations environment. This report includes statistics like data access rates, transaction volumes, and physical table sizes. The report also identifies data problems and resolutions.

3C.12.6.4 Proven Process

We employ a structured, disciplined approach for database design and implementation. This process is highly integrated with the system development effort.

The following subsections describe our process.

3C.12.6.4.1 High-level View

The database design is an integral part of the overall system design. As such, many database design and software component design activities overlap. Additionally, we develop and test software for accessing, modifying, and converting data, in parallel with the application build.

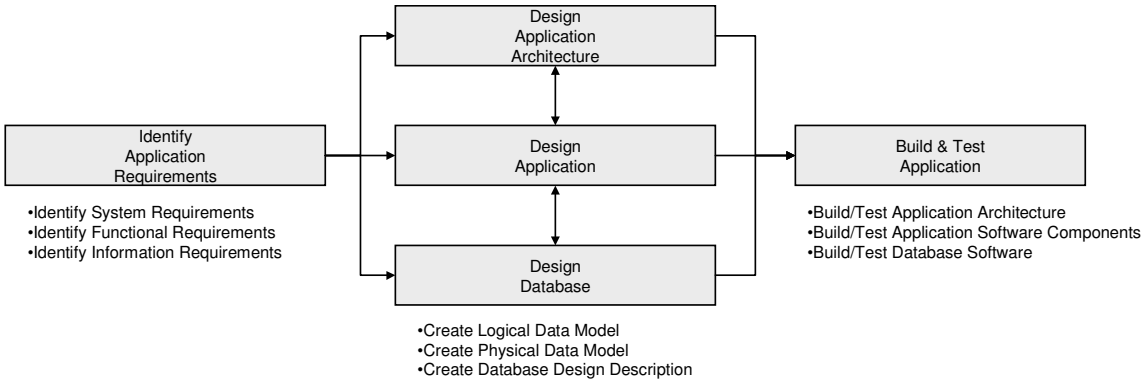


Figure 3C.12.6-9 Design Process – *The database design is an integral step in the overall design process.*

3C.12.6.4.2 Integration with System Development Team

Database design and development is an area that requires substantial interaction between the Database Development Team and the System Development Team. As shown in Figure 3C.12.6-10 System Development and Database Development Relationships, the project lifecycle has many points where the development team depends on the evolution of the database. The database team also depends on the functional expertise of the System Development team to review the evolving structure for functional soundness.

Each team is responsible for its deliverables. However, they collaborate on deliverable development. At the start of each development stage, the interaction between the two teams is more structured. Both system and database developers are involved in requirements gathering, conceptual design, technical design, database design, and test planning meetings. Each of these meetings marks the beginning of the evolution of a deliverable. After the information sharing has occurred, the individual designers/developers complete their deliverable. The loop closes at deliverable completion, when once again the teams come together to verify that deliverables are high quality in a Technical Review.

The teams also interact during deliverable composition, albeit in less structured ways. For example, detailed designs include data element information. If the data element does not exist, the software and database designers work together to define the new element to meet database standards and functional requirements.

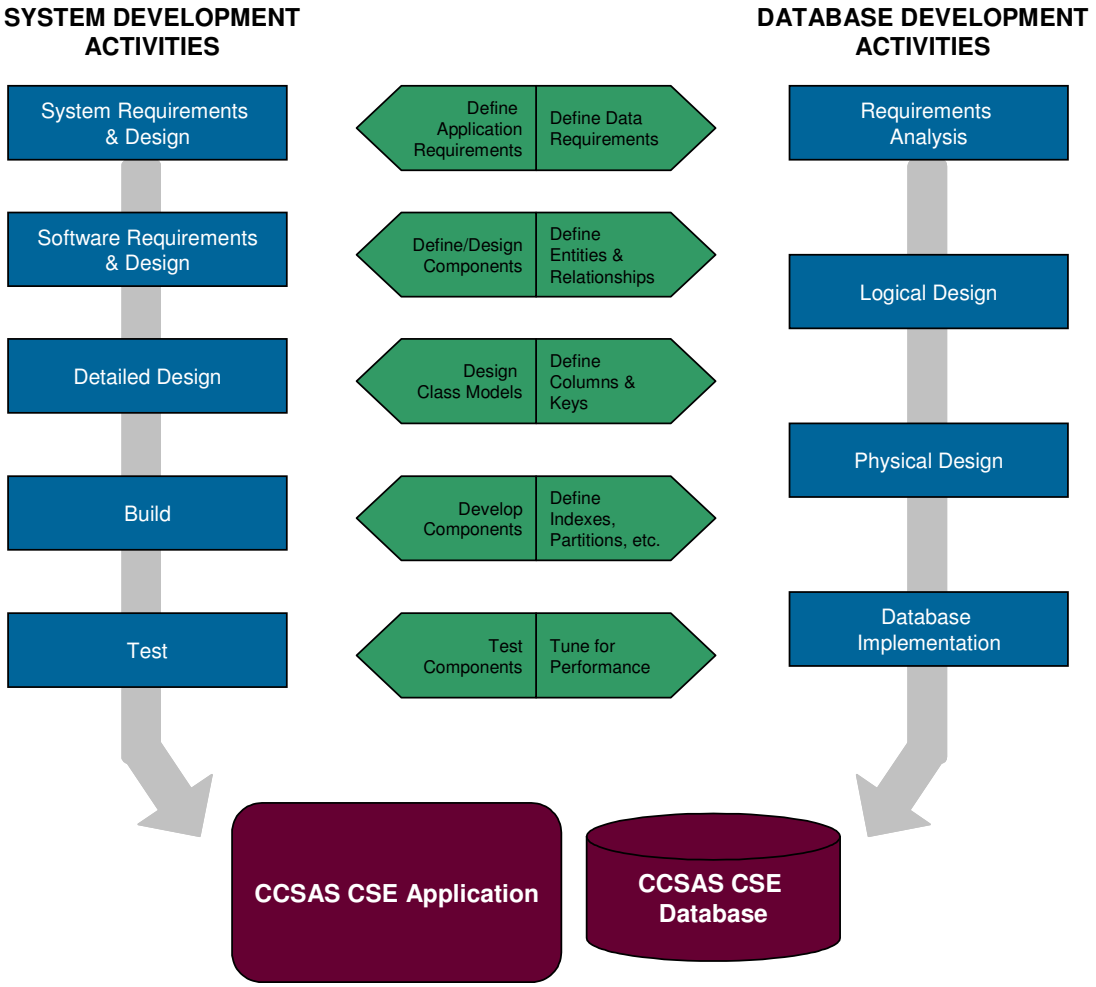


Figure 3C.12.6-10 System Development and Database Development Relationships – *The integration between the System Development and Database Development teams is critical to having a sound design and application.*

3C.12.6.4.3 Process Steps

As shown in Figure 3C.12.6-10 System Development and Database Development Relationships, the four steps in Database Development are:

1. Requirements Analysis
2. Logical Database Design
3. Physical Database Design
4. Database Implementation

These steps correspond to system development steps. The paragraphs below discuss the objectives of these steps and their interaction with the system development process.

3C.12.6.4.3.1 Requirements Analysis

Database requirements analysis occurs in parallel with requirements analysis and design. The system architects and database developers work together to begin forming many high-level structural decisions during this stage. For example: How will system entities (banks, employers, case members, payors, payees, etc.) and their demographic information (addresses, names, etc) be stored? What is a case and how does our structure support the flexibility to report a case per federal case construct vs. court system case definitions? Should the basis for accounting be court number-based (order), member-based, case-based, or other?

Database requirements analysis determines what data needs to be stored and how the system uses that data. Most data requirements of the CCSAS CSE database are gathered from the following sources:

- System input and output review. An important source of gathering data requirements is reviewing the system inputs and outputs. For example, the child support application shows a variety of types of information stored for case members. Reports, especially existing State and Federal reports (e.g., CS157, etc.), provide information about what the system should track and maintain.
- User Interviews. A primary source of requirements is user interviews. The value of involving users in this process is covered elsewhere in this document. User interviews provide an opportunity to understand the perspectives of different users. As with the case construct example, it is important to understand who uses the information and how they use that information. A few of the perspectives that we consider include IV-A agency interaction, federal reporting, court interaction, FTB collection, and DCSS oversight functions.
- Consideration of the legacy systems being replaced. Legacy systems should not be the basis for understanding data requirements, but can be used in two ways. First, legacy system review may uncover missing data requirements. Second, conversion from ARS and CASES may serve as a “tie breaker”. If two different approaches to storing data are equal in other ways, the ease of converting data should be considered.

- System Requirements. These requirements, documented in the System/Subsystem Specification, taken into consideration with the State's Base Business Concepts, influence database design.

We document and maintain database requirements in the:

- IEEE-based Software Requirements Specification (SRS)
- Logical Data Model
- Physical Data Model
- Software Design Description

Database requirement definitions include:

- Base storage requirements, like "store employer information such as employer name, employer identification number (EIN), physical address, payroll address, etc."
- Format and validation rules for specific fields, like "EIN is 9 digits, coming in as 12-3456789 but stored without the dash"
- Data access and usage, like "provide for employer lookup by EIN, name, name with wildcards, and phonetic name."

When data requirements change, we correspondingly modify the IEEE SRS, Logical Data Model, Physical Data Model, or Software Design Description. We do this in a controlled fashion, using our established change management procedures.

3C.12.6.4.3.2 Logical Database Design

The next step in database design is Logical Database Modeling. Logical design starts in conjunction with Software Requirements and Design and continues through Detailed Design of the application. The database starts to take shape during this phase as we define entities and their relationships with other entities.

We build and maintain a Logical Data Model (LDM). Modeling activities for the LDM include:

- Identifying and defining entities – name, description, unique identifier, list of attribute types, volume and growth information
- Identifying and defining relationships – name, entities involved, cardinality, description
- Identifying and defining entity attributes – name, description, data type, length, domain
- Developing a graphical representation of the LDM that depicts entities and relationships
- Reviewing the LDM
- Approval of the Logical Data Model

Logical database design defines the application's tables, the columns for those tables, the views used for accessing the tables, the referential integrity constraints, and codes tables to support the table columns. Section 3C.12.6.6 Proven Tools discusses the tools to complete this step. Logical Database Design includes a number of sub-steps:

1. Review aspects of the application that could affect the design of the database. There are a number of inputs into understanding the data requirements for the CCSAS database structure as described above in Requirements Analysis. It is important to keep these in mind.
2. Complete a conceptual model for the database using Entity Relationship (ER) diagrams. The ER diagram shows the entities and their relationships to other entities. It illustrates the database definition graphically.
3. Define the database. For the purposes of Logical Design, the database definition evolves through the following efforts:

    - Identify and define tables
        - Approach to handling entity-type hierarchies (options include all one table, separate tables for the supertype and each of the subtypes, or a combination of these approaches)
        - Approach to handling derived data (e.g., including it in the table or generating it at run time)
        - Approach to handling tables with excessive row lengths
    - Define table columns
        - Use of variable-length vs. fixed-length columns
        - Use of standard columns in every table (for example, time stamp and User ID for last update)
        - Use of nulls
        - Use of special data types (for example, DATE, Binary Large Objects)

- Define table primary key
- Identify foreign keys for each table. Relationships between entities are the foreign keys.
- Address Referential Integrity. For each foreign key, identify the column(s) that it includes, identify the table that is being referenced, and identify the referential integrity rules to be supported by the DBMS. The following situations should be addressed:
  - Insertions involving a foreign key
  - Updates to a primary key of the referenced table
  - Deletions from a primary key of the referenced table
- Define database views. Views can simplify a program's or report's view of data, as a security mechanism, or to help isolate programs and end users from logical database design changes.

4. Denormalize the database design when there is strong evidence that there may be a performance problem. Denormalization techniques are discussed elsewhere in this document.

After denormalizing the design, it may be necessary to reconsider referential integrity constraints. An alternative is to defer the implementation of referential integrity until after having made initial denormalization decisions.

5. Conduct the informal reviews regarding the logical database design with database administrators, application developers, and appropriate CCSAS project decision makers.

   Component designs and code will depend on the logical database design. To avoid later rework, program designers need to understand and agree with the logical database design. End users should also review the design to check that their needs are being met for different processing environments (e.g., case processing, transaction processing, decision support).

We start Logical Database Design during Software Requirements Analysis and Software Architecture Design. We complete it during Software Detailed Design. A key work product is the Class to Database Mapping of Software Detailed Design. This ties together the class model with the database model. See Section 3C.12.5 Software Detailed Design for more detail on the Class to Database Mapping work product.

3C.12.6.4.3.3 Physical Database Design

Next, we design the Physical Database Model. We begin this step during Software Detailed Design. We continue to evolve the PDM during the Software Coding.

The Physical Database Design is a blueprint for how to store the data for the CCSAS CSE system. The main consideration for the Physical Design is performance. Performance is improved through denormalization and data allocation. The “Proven

Tools" section in this document describes the tools used to complete the PDM. Physical Database Design includes a number of sub-steps:

1. Design internal storage structures, including tablespaces, indexes, and buffers.
2. Design indexes. This is a significant physical database design activity. Index characteristics to consider are column order, storage scheme (sorted, hashed), and uniqueness. Indexes are designed to support data access requirements while reducing physical input/outputs (I/O). In general, indexes improve retrieval performance but add overhead to insert, update and delete operations. They also require additional storage. The database designer evaluates these trade-offs.
3. Consider other factors for performance requirements, where known areas of concern exist. Such considerations include:
   - Selecting the physical sequence of data
   - Deciding how tables are grouped in physical storage
   - Using tablespaces and databases to allow for the grouping of tables
   - Partitioning a table based on ranges of key values, and specifying different physical storage options for each partition
   - Allocating free space to allow for future growth
   - Specifying the size and duration of locks
   - Allocating buffer sizes
   - Assigning the underlying files to physical storage devices
   - Deciding on the frequency of backup and reorganization activities (this involves mapping the overall strategy defined in the application infrastructure to the individual components in the database design)
   - Deciding on techniques for achieving high data availability (for example, replication, mirroring)

The database designer may alter decisions made during logical database design and may perform further denormalization.

4. Deliver a Physical Data Model (PDM). We transform the LDM, which contains table names, field names, and primary/foreign keys, into the PDM. The PDM is a ready-to-be-implemented model, as it additionally includes field types, storage parameters, and indexes.
5. Conduct the informal reviews regarding the physical database design , with appropriate parties.

3C.12.6.4.3.4 Database Implementation

We begin database implementation[36] shortly before or early in the Software Coding phase of System Development. We continue these tasks throughout maintenance and operations. During this time, the components of the system are developed and tested against the database. Some of the activities that are included in the Database Implementation phase include:

[36] The use of "implementation" here should not be confused with the full-fledged implementation, or roll-out, effort.

- Implementing the physical database. This is achieved through the use of data definition language (DDL) scripts. We track these scripts as discussed in Section 3C.12.6.5.1 Making Changes and Analyzing the Impact.
- Tuning the physical database. Based on performance needs, this tuning is an ongoing process. The database designer should pay particular attention to the performance of the following programs:
  - Programs that access the largest tables
  - High-volume programs that process a significant amount of data
  - Complex programs
  - Programs that access data without using the primary key
- Monitoring database activity. This is an ongoing function of the database team. The Database Activity Report is the reporting mechanism for an ongoing Database Activity.

3C.12.6.5 Proven Standards and Reviews

We use standards for database schema, metadata, and SQL that are based on lessons learned and proven practices from previous engagements. We reference our experience of previous child support engagements to develop naming and other functional standards. We reference large Internet, component-based engagements for technical standards. Naming standards are boring, and enforcement is unpopular work. However, good and consistent use brings years of payback in maintainability.

We conduct formal reviews of the enterprise-wide data model as well as the logical and physical models. In addition to the formal reviews, informal reviews happen periodically during the project. These provide opportunities to share information across teams.

3C.12.6.5.1 Making Changes and Analyzing the Impact of Changes

Regardless of the selected tools for data modeling and metadata modeling, we follow documented procedures for making changes and analyzing the impact of changes.

We version our models using procedures and tools that resemble those for versioning application models. We also use version control on any related DDL scripts, database alteration scripts, and operating system configuration/alteration scripts.

We version database designs, just as we version application designs.

In addition to integrating version control, we integrate database development with requirements and testing. We do this with tool support, if available, and/or with additional utilities and procedures.

As well, we store the entire DB schema within the tool. This means that DDL creation and alteration scripts can be generated from information provided by the tool. We keep the design model and actual implementation synchronized, thereby avoiding out-dated designs.

3C.12.6.6 Proven Tools

3C.12.6.6.1 Integrating Software and the RDBMS

Fundamentally, SQL (Structured Query Language) is the connection between business logic in applications and table definitions in the database. We build the CCSAS online and batch applications in Java. This gives us two primary approaches for integrating Java objects with relational tables:

- JDBC. Java Database Connectivity (JDBC) is a J2EE standard for embedding dynamic SQL in Java. JDBC is an Application Programming Interface (API) for database-independent connectivity between the J2EE platform and a wide range of relational data sources. A JDBC driver allows Java objects to: perform database connection and authentication, pool connections for performance (transparently to the programmer), manage transactions, move SQL statements to a database engine for preprocessing and execution, inspect the results from SQL statements, and execute stored procedures. JDBC is considered dynamic, because the JDBC driver translates the SQL strings at run time.
- SQLJ. SQL in Java (SQLJ) is an American National Standards Institute (ANSI) standard for embedding static SQL in Java. SQLJ was developed to complement the dynamic JDBC SQL model with a static SQL model. Like JDBC, SQLJ allows a Java application to perform connection and authentication to a database server, manage transactions, execute stored procedures, execute SQL statements, and handle results. Unlike JDBC, SQLJ is considered static, because an SQL programmer uses a precompiler at development time to convert the SQL strings into lower-level statements that are executed at run time. COBOL developers who are accustomed to embedded SQL surrounded by “EXEC-SQL” and “END-EXEC” statements will appreciate SQLJ’s similar syntax.

IBM technology supports both JDBC and SQLJ access from WebSphere, our Java application server, to DB2, our relational database. Section 4A.4 CSE System Environments discusses IBM WebSphere and IBM DB2.

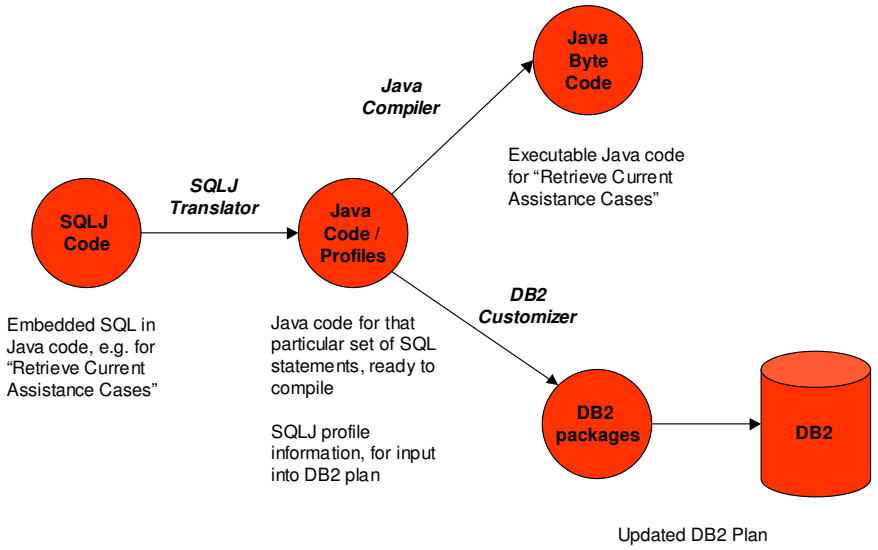


Figure 3C.12.6-11 SQLJ – *SQLJ uses a straightforward syntax that is compiled, thereby providing compile-time validation and improving performance.*

We strongly consider SQLJ rather than JDBC for application correctness, developer productivity, and performance:

- Correctness. SQLJ's precompilation step validates table names, column names, and database types. SQL programmers can therefore detect errors, like typographical errors, misnamed columns or mismatched types, at compile time. With JDBC, SQL programmers do not detect errors until run time. This delayed detection is typically more expensive.
- Productivity. SQLJ syntax is more natural and intuitive. Developers embed the SQL statement directly in the Java. They do not worry themselves with the additional details of more complex API calls, as with JDBC. Moreover, SQLJ uses a host variable approach for handling input parameters and result data. This contrasts with the tedious and error-prone JDBC approach, which uses result sets and requires steps to pack and unpack data.
- **Performance.** IBM's SQLJ precompiler creates a plan in the database.[37] This is a fundamental advantage of the static approach, as it avoids the "prepare" step required in the dynamic approach. Letting the database know about SQL statements in advance has two other benefits. First, the database optimizer capitalizes on this compile-time information to improve run-time performance. Second, we can use the "explain" statement to determine how the database executes an SQL statement – whether it uses an index scan, a table scan, etc. The SQL developer reviews this explanation with a DBA, to discuss improvements to the SQL and the database schema.

SQLJ's precompilation links Java application code with the physical database schema. This integrates software and database development.

[37] SQLJ precompilers from different vendors work differently, and therefore perform differently.

3C.12.6.6.2 Data Modeling

We use the data modeling tool, to define:

- Databases and tables
- Columns with datatypes, defaults, domain constraints, uniqueness, and nullability
- Primary keys, foreign keys, and indexes.
- User-defined information to be stored directly in the model, such as volumetrics. This can include access characteristics and frequency of use.

The project's data modeling tool will support multiple databases that support the SQL-92 standard. The DDL (Database Definition Language) programming language particular to these databases will be employed. This allows project personnel to generate database schemas, including not only table and column names, but also keys, indexes, and other constraints.

3C.12.6.6.3 ER*win* Examiner for Quality Reviews

We use ER*win* Examiner to help automate quality reviews.[38] Examiner is an expert system that validates database structures for integrity and performance. It checks the schema for deviations such as inconsistent column definitions, incorrect functional dependencies, infinite cycles, incorrect relationships, unnecessary indexes, or necessary but undefined indexes.

[38] ER*win* Examiner is for quality reviews and should not be confused with ER*win's* database modeling tool.

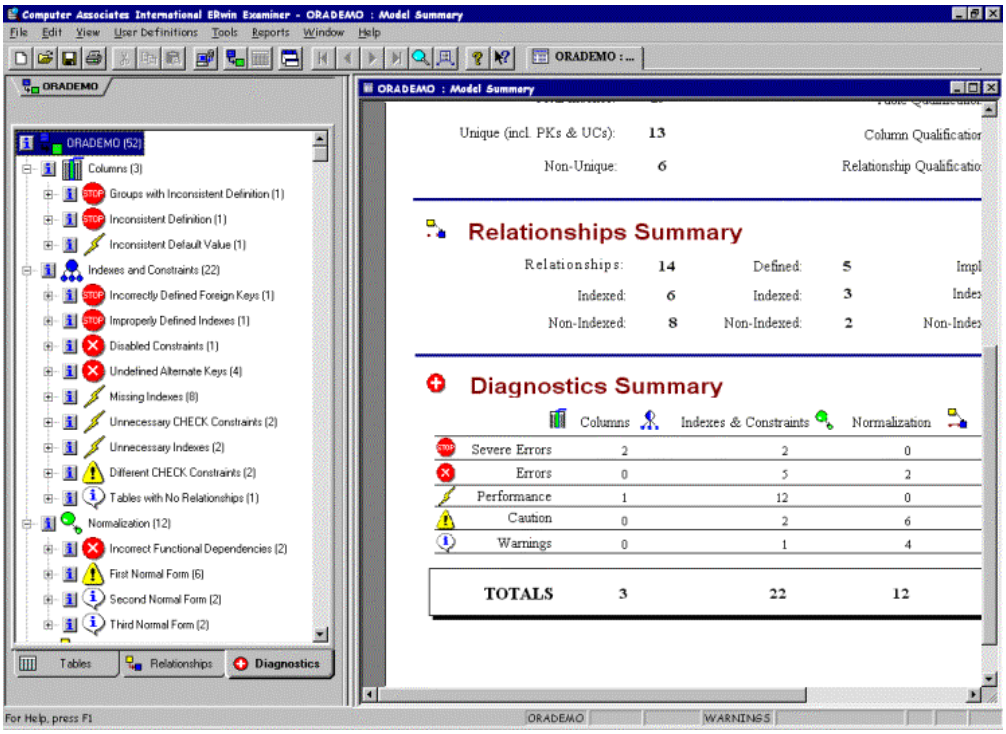


Figure 3C.12.6-12 ERwin Examiner - *Examiner highlights potential problems with integrity and performance in the schema.*

Examiner both detects problems and suggests improvements. In several cases, it can automatically generate the DDL change scripts.

3C.12.6.7 Conclusion

Our application relies on a highly reusable, well-factored object model. While the data in these objects needs to be stored in the database, the physical data model does not map directly to the object model. Denormalization and other performance concerns prevent us from a having a simple, one-to-one mapping.

This makes our database design even more important. With a traditional client/server system, designers hand-crafted data structures for each individual transaction. With an object-oriented approach, we must now figure out how to take the individual transaction data and put it into a reusable business object. The alliance brings to bear years of proven experience with difficult challenges like these.

Our database design meets today's needs and evolves as child support legislation and requirements change.

We also bring a disciplined approach to database design and development. One that includes up-front planning, clearly defined tasks and phases, collaboration between application and database designers, market leading tools, and a toolkit of proven approaches and principled techniques.

3C.12.7 Software Coding

Building maintainable code is much more than attending a "Java in 21 Days" class, buying a Java IDE, and following the J2EE specification. It requires craftsmen and women who have mastered advanced, proven object-oriented techniques. It requires a structured approach for predictable delivery, and reuse calls for planned collaboration. Alliance developers bring this know-how to CCSAS.

3C.12.7.1 Introduction

During the coding phase, the alliance delivers high-quality software modules that:

- Meet business requirements
- Scale for high performance and throughput
- Can be modified or extended in a straightforward fashion by future maintenance teams (staffed by the alliance or otherwise)
- Support "fix in one place" by centralizing common, shared logic
- Adapt to changes in requirements or business rules (e.g. prompted by new legislation)
- Standardize child support practices across the counties while also allowing for county-specific requirements
- Integrate well with outside systems, both current and planned
- Follow industry, open standards

A prerequisite of high quality code is high quality design, traceable to solid business requirements. The points above already have been addressed in Section 3C.12.5 Software Detailed Design. However, quality design alone does not guarantee quality code. We apply additional techniques and proven approaches during the coding phase for maintainability and extensibility. And we do this in a controlled fashion, using leading tools and established procedures as we edit, compile, and debug.

Our coding techniques result in readability and maintainability. Our coding procedures result in controlled, measurable progress.

3C.12.7.2 Proven Principles

Methods, approaches, and procedures are absolutely necessary for delivering code in a disciplined, engineered fashion. But methods alone do not result in delivering quality code – code that is maintainable, adaptable, and extensible. Delivering quality code requires proven techniques, principles, and standards, which the alliance brings to the CCSAS project. Many of these techniques have already been addressed up-stream, during design. Principles that are specific to the coding effort include:

Our coders write for maintenance teams, years into the future.

1. Prototype with multiple reference applications
2. Use the same programming language for online and batch
3. Make version management just a part of the daily routine
4. Integrate regularly during development
5. Use inheritance judiciously
6. Use delegation and aggregation aggressively
7. Use Java interfaces liberally
8. Code to prevent misuse
9. Improve readability with comprehensive standards

3C.12.7.2.1 Prototype with Multiple Reference Applications[39]

We believe our use of multiple reference applications is a key differentiator of our approach. A reference application is a "slice" of the CCSAS CSE application, created for:

- Proving out specific aspects of the technical architecture
- Proving out our development methodology, tools, and procedures
- Acclimating our application developers to the CCSAS CSE-specific development environment and APIs. This is a key part of knowledge management
- Providing a demo for key CCSAS CSE project staff and users. This is an important, "tangible manifestation of progress"
-

Fundamentally, we use a reference application to mitigate risk and get developers up to speed on the CCSAS CSE architecture.

The technical architecture team builds the architecture and shells of reference applications for online, batch, and eAI platforms. Application developers then build upon these shells with application-specific logic.

We build these reference applications at the beginning of application development efforts. This supports quickly resolving issues with the architecture as well as the development environment.

[39] The Reference Application is primarily an architectural exercise, proving out architectural approaches planned for use on the project.

3C.12.7.2.2 Use the Same Programming Language for Online and Batch

We use Java for both online and batch development. This has the benefits of:

- Allowing developers with skills in one programming language to be immediately productive in both environments. This also supports better load balancing of resources from a project management perspective
- Reusing the same development environment, including tools, standards, and guidelines
- Reusing the same coding processes (like "Fast Track", introduced in Section 3C.12.5 Software Detailed Design)
- Reusing a similar execution and operations environment
- Reusing similar technical services for error handling, logging, auditing, and other functionality. We use the GRNDS frameworks, introduced in Section 3C.12.4 Software Architecture Design

Java's performance has improved significantly over the past few years, to the point where it runs comparably to C++. It is therefore an appropriate solution for batch from a performance standpoint, as well.

Accenture brings to the alliance its intimate understanding of batch architectures it delivered on other large, public sector projects, which have used similar technologies. We anticipate having access to relevant assets from these federally funded programs.

3C.12.7.2.3 Make Version Management Just a Part of the Daily Routine

Coding in a disciplined manner requires frequent use of version control. Our developers check-in interim versions regularly – not just at major checkpoints. This keeps them focused on delivering functionality in small increments, which improves productivity. Frequent versioning also better supports reverting to previous versions, if necessary to accommodate changes.

Version early and version often.

Alliance coders each have a license for the version management tool. This contrasts with a shared license model, in which:

1. Developer Julie asks Jim, who has a license, to check out Julie's module.
2. Julie makes multiple revisions of her module and manages them by herself (e.g. with a personal folder system to temporarily store previous versions).
3. At major integration points, Julie asks Jim to check her module back into the repository.

A component-based environment, however, needs different approaches to version management. Our developers have access to the version control tool, integrated into the IDE, because:

- A single developer is responsible for managing dozens of dependent classes simultaneously. A personal folder system does not work well. In other words, the developer needs access to tools to help with version management.
- Classes are reusable across developers and across work cells. Because of this interdependency, the repository should always be as up-to-date as possible. We have policies for frequent check-ins, as mentioned, as well as regular builds.

One of the costs of reuse is that configuration management is harder on object projects. We use appropriate tools – in the hands of developers, who own the versioning of their classes – to manage this complexity. Moreover, some application developers dedicate part of their work effort as build masters.

Section 3C.4 Configuration Management addresses the actual version management tool and procedures.

3C.12.7.2.4 Integrate Regularly During Development

To manage and control interacting software modules, we continuously integrate completed code into the overall system. (We use the phrase “continuous integration” to follow industry parlance. Readers should interpret “continuously” in this context to mean periodically, rather than at every single moment.)

This contrasts with the traditional approach to integration: waiting to incorporate modules until the end of a coding phase. With the traditional approach, the team promotes a collection of modules together, expecting to create a unified build that is then ready for Unit Integration and System testing. This approach can work for applications designed in a procedural, “stove-pipe” fashion. When modules have minimal dependencies, compilation and linkage problems are less likely in a software build.

But the traditional approach falls apart with component-based development. First, an individual coder is now responsible for simultaneously managing dozens of dependent, small-grained classes. These intertwined modules can quickly become unsynchronized unless versioning is part of habitual coding activities, rather than an after-thought. Second, classes are reusable across developers and across work cells. This interdependency calls for up-to-date software builds, as developers complete code modules, before unit integration testing officially begins.

We appreciate that object-oriented projects introduce the challenge of managing hundreds of separate class modules. In many ways, configuration management is harder than with traditional, procedural development. We handle this challenge with successful policies for frequent versioning and builds.

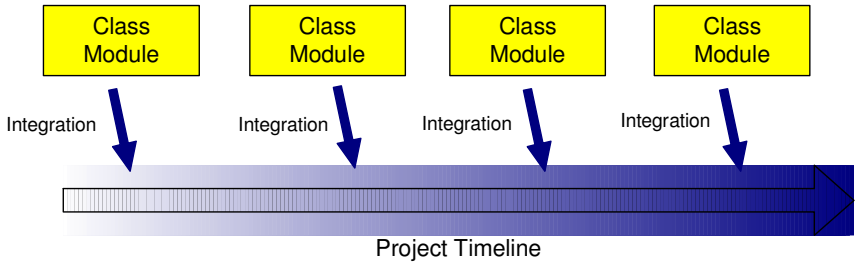


Figure 3C.12.7-1 Continuous Integration - *Continuous integration improves the quality – and timeliness – of a software build.*

After finishing a code assignment, a developer tests the code in the context of the up-to-date build. We avoid the cost of redundant testing by using automated unit tests, as described in Section 3C.12.9 Test Management.

Our methodology therefore manages the incorporation of software components into the build. This approach contrasts with more traditional approaches which only attempt software integration after dozens developers have built multiple modules over several week or months. By deferring the incorporation of these components into an up-to-date build, the traditional approach presents a greater configuration management challenge in that hundreds of modules are introduced into the configuration at once. In contrast, our methodology manages the smooth introduction of new software code into the configuration, on a regular basis. Using smaller change sets, due to frequent builds, makes it easier to isolate and address problems earlier in development.

New techniques for integrating modules into builds improve configuration management.

To the extent practical, the alliance structures coding tasks into small reusable components that can be coded and unit tested within a relatively short period of time. In this manner, we avoid large amounts of "work in progress." As well, a particular build will have incorporated the latest and therefore more closely approximates the current status of development.

Continuous integration does not change formal version control, configuration management, or build release processes. We still exercise the same level of care, follow the same processes, apply the same version management standards, and use the same tools.[40] Rather, continuous integration is simply a way to make sure software modules work with each other on a just-in-time basis **during development.** This detects build problems sooner, rather than later. This addresses, in a robust way, the IEEE requirements in clause 5.4.1 that relate to coding.

3C.12.7.2.5 Use Inheritance Judiciously

[40] See Section 3C.4 Configuration Management.

Typical component-based development efforts use encapsulation, polymorphism, and inheritance – techniques first introduced by object technology. Many so-called object experts naïvely apply these techniques, whereas our developers:

- Know how to properly apply traditional and newer techniques
- Understand the relative value and appropriate context for each technique

Consider inheritance, which is the easiest object-oriented practice to learn and use. Developers use inheritance for new classes to extend the code base of existing classes. In object-oriented programming languages, having one class "inherit" from another is a straightforward, one-line statement.

Yet inheritance does not provide as much benefit as other techniques like encapsulation and polymorphism. It is also the easiest to overuse. For example, inheritance hierarchies, which define how classes reuse each other's data and methods, can quickly become too intertwined. Overly complex hierarchies are difficult for maintenance developers to understand. They also complicate impact analysis.

> Inheritance is easy to use…and *over-*use.

Because inheritance ties classes to each other in a static way, intertwined hierarchies are brittle to changes and ripple impact effects.

3C.12.7.2.6 Use Delegation and Aggregation Aggressively

We have experience with and mastery of newer – and also proven – techniques. In place of complex inheritance relationships, we often favor delegation and aggregation. They work as follows:

- Inheritance: the subclass of the object calls one of potentially many inherited superclasses to accomplish work
- Delegation: the object instead calls a separate helper object
- Aggregation: the object instead calls a contained helper object

Inheritance is a static technique: developers specify in advance which code to reuse. Inheriting a superclass can require the developer to implement certain methods, even if they are irrelevant to the particular subclass.

In contrast, both delegation and aggregation are dynamic techniques. The code that is reused can be changed, if necessary, at run time. As well, these techniques can better distribute responsibilities between distinct objects, keeping a particular class from becoming too intelligent, or monolithic.[41] Both techniques provide a similar level of code reuse as inheritance, without the problems associated with brittle, static hierarchies.

> With delegation and aggregation, we support reuse in a more flexible way.

[41] The code quality tool, mentioned in Section 3C.12.7.5.3, helps detect complex inheritance hierarchies as well as classes that are "too intelligent."

These techniques are more sophisticated: developers cannot simply read them in a book. Team members need to have prior experience. The alliance brings such experience to the CCSAS project.

This does not mean we avoid inheritance entirely. Rather, we use it appropriately, while also applying other techniques.

3C.12.7.2.7 Use Java Interfaces Liberally

More experienced coders also use interfaces: an advanced object-oriented technique that complements encapsulation. An interface is an explicit representation, in the software, of a set of related methods that are available to the outside world.[42] A component is distinguished from an object in that its interface (what the component does for the outside world) is separate from its implementation (how the component does work internally, in the source code).

Interfaces allow programmers to better hide the internal details of a component. They reduce the coupling between different components, which results in more modular, maintainable software.

Multiple interfaces provide multiple views into the same reusable class.

This is because interfaces go a step beyond the "public" designation for methods. An interface groups together a *logically related* subset of public methods. Consider a **Non_Custodial_Parent** class, which might have 40 public methods out of 100 total methods. Maintainable code might separate out logically related methods for establishment functionality (e.g., blood sample information, verification codes) from methods for enforcement functionality (e.g., employer, wage garnishing, and financial information). Moreover, another interface might contain methods related exclusively to database access for this object.

This decomposes a complex class into multiple, simple interfaces. Interfaces on the same object are analogous to database views on the same table.

[42] Use of "interface" in this manner refers to an Application Programming Interface (API) rather than a Graphical User Interface (GUI). Recall Section 3C.12.4 Software Architecture, which explains this difference.

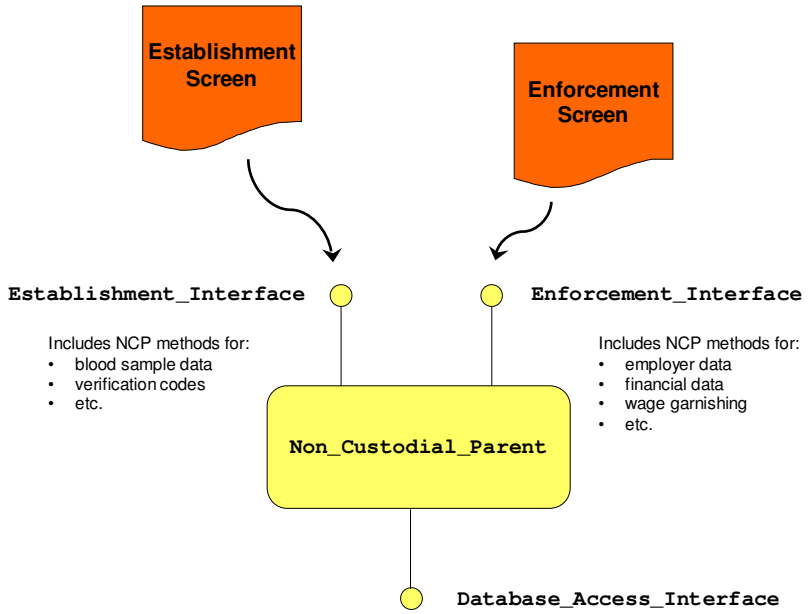


Figure 3C.12.7-2 Multiple Interfaces - *Different developers can look at the same class in different ways.*

Using multiple interfaces has the following benefits:

1. In the reusable object model perspective, a single class represents non-custodial parents, thereby supporting "fix in one place." But at the same time
2. Developers reusing that class need not concern themselves with the details of the entire class. They worry only about the interfaces related to their particular transaction. This improves modularity

In other words, interfaces gain the cohesiveness of point (1) but also the loose coupling of point (2).

Use of interfaces is more than just a tool to decompose and organize code. It physically separates the client code from the server code, because the client binds to an interface rather than a particular implementation (i.e. the executable code). This allows us to address multiple architecture attributes desirable to the State:

- Physically distribute the component to a different machine without having to re-compile or re-link the client. This supports the architecture attribute of portability.
- Make changes to an existing implementation without having to re-compile or re-link the client (assuming that the interfaces have not changed). This supports the architecture attribute of maintainability.

Separating the interface from the implementation produces more maintainable, modular, and portable software.

- Replace an existing implementation with a totally new one, custom or COTS. This supports the architecture attribute of modularity and adaptability.

Lastly, Java interfaces also map directly to CORBA and even DCOM constructs[43] and mechanisms. These platforms are not within the current business requirements, and they are outside the scope of this contract. But future requirements, like integration with statewide e-Government intake functionality, might require hosting our components in these heterogeneous environments. We deliver extensible code than can be gracefully migrated if necessary, while minimizing additional, future work. This supports the architectural attribute of interoperability.

3C.12.7.2.8 Code to Prevent Misuse

Components and classes are intended to be reusable, in unpredictable ways. We code the software in a robust way, so that its eventual reuse is consistent with its original design. Clear, well-documented Java interfaces lead to proper use. In addition, we code in extensive error handling to prevent *mis*use.

Our GRNDS architecture, described in Section 3C.12.4 Software Architecture Design, includes a production-tested, error-handling framework. But powerful error handling involves much more than just a mechanism for the application to signal errors and display messages to the user. Our developers also leverage:

- Proven guidelines. These specify how to define errors, when to raise them, and how to handle them within the different layers of a distributed application.
- Typed exceptions.[44] This technique forces calling code to understand, at compilation time, which errors it must be prepared for. This makes it easier for developers to catch problems earlier in the development process.

Error handling cannot be an afterthought. Our techniques result in robust software.

- Pre-conditions and post-conditions. Pre-conditions prevent the execution of a routine unless explicit entry criteria are met. Post-conditions catch any inconsistencies with a routine's exit criteria. We use a GRNDS framework for building pre- and post-conditions into the software. This detects programming errors at their source.

3C.12.7.2.9 Improve Readability with Comprehensive Standards

To address TM 12.7.1, we also use proven coding standards. Our Java, HTML, and SQL coding standards are:

[43] Like the standardized Interface Definition Language (IDL) of the Object Management Group (OMG), which is responsible for CORBA, or the Microsoft Interface Definition Language (MIDL) of Microsoft, which is responsible for DCOM.
[44] A *typed exception* is a Java term for an error that the compiler can recognize.

Standard # 2.3.4: *whitespacedoes,ingeneral,enhancereadability*

- Comprehensive, with more than just pretty indentation rules or ways to use a "while" loop. They also include more subtle – yet more important – guidelines for making code readable. Like limiting the number of statements in a method or re-factoring complex if-then-else conditions.
- Well-established, coming from years of experience with literally hundreds of developers. Moreover, our Java standards have leveraged some of the best thinking from our Smalltalk and C++ standards.
- Well-documented, which encourages actual usage from developers
- Derived from published and well-known standards

In our experience, even the highest quality standards are ignored if they are published in a binder and left on developers' desks. Instead, we host formal training to introduce new developers to our standards. Helping them understand the rationale for a particular standard builds their ownership in that standard.

Code maintainability is affected not only by the actual code instructions but also by the corresponding code comments. We focus on the "softer side" of coding with proven commenting standards. As expected, coders write in-line, embedded Java comments. But as a value-add, we also deliver JavaDoc, which explains classes and methods in readable HTML format. JavaDoc supports a higher-level view of the code, which can be useful for maintenance developers and others. Experience has shown that this higher-level view increases understanding. It is hence more likely to result in appropriate bug fixes and enhancements. See Section 3C.12.5 Software Detailed Design for more detail on JavaDoc.

Coding practices and standards lead to consistent code, which helps State personnel learn the CSE software. This facilitates knowledge management.

Alliance code is self-documenting. This is not a simple matter of following coding standards. Coders need skills for representing a complex set of instructions in a simple, readable fashion. The alliance brings these skills to the CCSAS project.

3C.12.7.3 Proven Work Products

3C.12.7.3.1 Inputs

For coding activities, we continue the "Fast Track" detailed design and coding process. See Section 3C.12.5 Software Detailed Design.

Our detailed designs contain the design artifacts that are inputs to this workflow. Key artifacts include:

- Method signatures for Java classes
- UML use case realization diagrams (carried over from design)
- HTML layouts

Coding starts with, and later updates, detailed design work

- HTML conversation flows

We use the Java code stubs created in detailed design to start our software coding activities. These code stubs include class definitions and important method signatures, including the method name, parameters, return value, and exceptions for each such method. In this manner, we base coding upon the software detailed design.

For entry criteria into coding activities, these design artifacts have been fully specified and quality reviewed.

3C.12.7.3.2 Outputs

Section 3C.12.2 System Architecture shows how CCSAS CSE follows fundamental layering principles. These principles separate user interfaces, business logic, database access, and integration logic. The following tables show how we separate our work products by architecture layer:

Client work products for the user interface layer include:	
Java Server Pages (JSPs)	HTML web pages with dynamic Java content
Table of user error messages	And their typical causes
Java applets	A complementary alternative to JSPs for interactive content; these will be used in a very limited fashion where appropriate

Work products for the business logic layer include:	
Java source code	That conforms to the industry standard Java 2 Enterprise Edition (J2EE) specification as well as our own coding standards
Java object code	The compiled (and executable) form of that source code
J2EE deployment descriptors	Specify how the Java object code should be run in the J2EE application server
JavaDoc	Additional supporting documentation in HTML that explains each Java class and method; JavaDoc helps maintenance developers understand code at a higher level (this adds value beyond typical, in-line code comments)
Updated Unified Modeling Language (UML) diagrams	These outputs of the design phase are updated, as necessary, as coding progresses

Work products for the database access layer include:	
SQL statements	That are embedded in Java, using SQLJ[45] to access the database
Stored procedures	Where appropriate
Database Definition Language (DDL) scripts	Fully specify table layouts in the database
Updated Entity Relationship (ER) diagrams	These outputs of the design phase are updated, as necessary, as coding progresses

Work products for the messaging layer include:	
Message formats	In industry standard XML
Format and validation rules	Documentation, in IEEE-based Interface Design Descriptions (IDDs), that explains the format and validation rules for these messages, for agencies that integrate with CCSAS CSE
Message sending and receiving logic	That conforms to the bindings of the eAI bus or to the industry standard Java Messaging Service (JMS) specification
Validation logic	In industry standard XSL, which checks if incoming and outgoing messages can be processed
Transformation logic	In industry standard XSL as well as Java, which translates from our XML standard formats into legacy, proprietary formats

We build out the messaging layer, and the messaging logic it contains, with eAI technologies and approaches. See Section 3C.12.8 System Integration.

Finally, work products across layers include:

[45] Section 3C.12.6 Database Design and Development discusses SQLJ.

Finally, work products across layers include:	
JUnit test scripts	Automate unit testing of business logic. This value-add dramatically improves on the traditional approach to unit testing, where developers create disposable, ad hoc scripts. JUnit scripts introduce regression testing at the unit testing level, and these scripts follow a standard
JUnit test suites	Group JUnit scripts so a particular application build can be regression tested with a single statement (effectively "at the click of a button")
Code review checklist	For a group of related modules
Build scripts	Package together development artifacts into a particular release

3C.12.7.4 Proven Process

Many companies offer trendy object-oriented techniques. What we offer that they may not are techniques applied with structure and discipline. We use proven approaches and methods for managing the development of software components.

In particular, we:

- Continue the alliance's "Fast Track" detailed design approach into coding. This minimizes shelf ware documentation and improves traceability between design and code.
- Continue using RUP artifacts into coding, in ways that are consistent with IEEE standards
- Use market-leading tools for the coding effort
- Expect quality and inspect quality along the way

Our forward-thinking approach is grounded in a structured-development heritage, which has been honed over the past two decades.

- Our management processes use earned value metrics. Code development proceeds in a controlled fashion, according to schedule.

3C.12.7.4.1 Roles

Key roles involved in coding activities are:

- **Application Developer,** who writes the code and unit tests according to the detailed designs. The developer also authors any new documentation (like JavaDoc) and updates existing design documentation. A developer reports into a Work Cell Lead, mentioned below.
- **Coding Supervisor,** who reviews code, suggests applicable coding patterns and techniques, and helps coders with debugging problems. A supervisor also delivers more complex code modules. A supervisor reports into a Work Cell Lead, mentioned below.

- **Frameworks Team Personnel,** who help application programmers use frameworks (like GRNDS; see Section 3C.12.4 Software Architecture Design) and other architecture services. These personnel report into the Technical Manager, a role that is specified in the Project Management Plan (PMP) deliverable (PM 001).
- **Work Cell Lead,** who manages the coding tasks on the work plan, e.g. schedule, budget, earn-versus-burn metrics, and sign-off. A lead reports into the Application Development Manager (also in the PMP).

3C.12.7.4.2 Process Steps

Recall that "Fast Track" includes the following high-level steps:

1. Kick-Off Meeting and Issues Resolution
2. Detailed Design Meetings
3. Documenting the Detailed Design
4. Interim Design Reviews
5. Code and Code Review

Section 3C.12.5 Detailed Design has already addressed the first four steps. The following paragraphs explain the code and code review steps.

3C.12.7.4.2.1 Code

A primary output of detailed design activities are the Java class definitions. These definitions, or shells, includethe definition of the class, its operations, and its attributes.

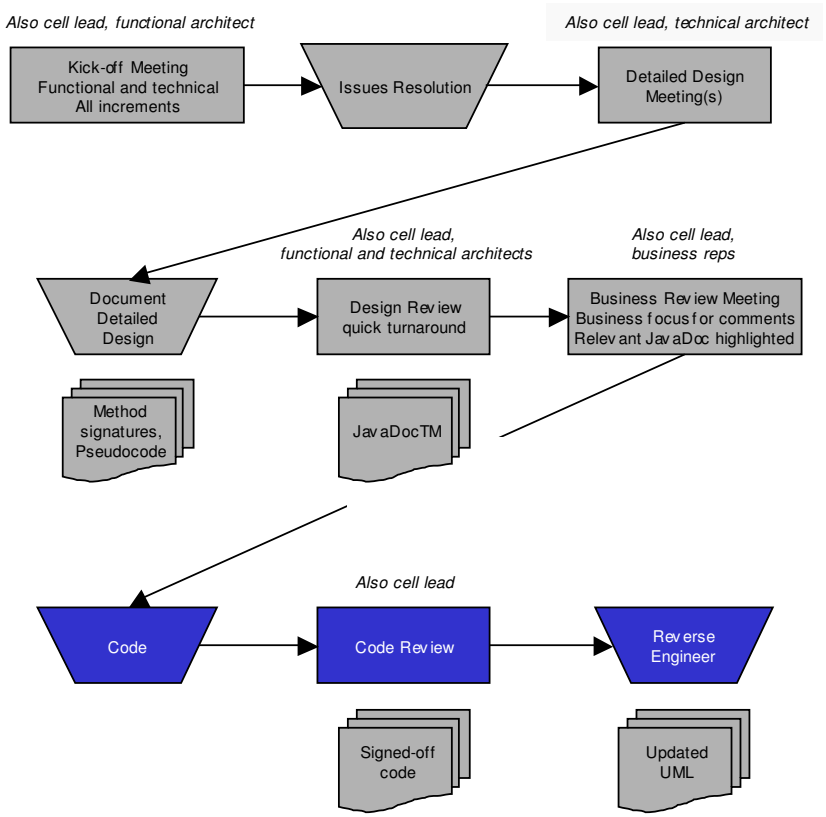


Figure 3C.12.7-3 "Fast Track" Coding Steps - *Fast Track continues from detailed design into coding.*

From these shells – that is, from the specified method signatures – the developer fills in the blanks. A shell already specifies the inputs, outputs, and exceptions for each public method. In completing the lines of code, the developer uses the class' attributes (internal data) that has also been captured during detailed design meetings.

It is important to understand that an object-oriented "method" is typically of a smaller granularity than a procedural "function." In other words, the developer writes fewer lines of code to fill in a method than fill in a function. This makes coding a more straightforward task in object-oriented development, because we have already performed much of the software engineering (and decomposition) during detailed design.

During the coding process step, developers use the proven practices described earlier in this document. They follow published coding and commenting standards.

3C.12.7.4.2.2 Review Code

We conduct technical reviews for Java code, JavaDoc, and HTML. Skilled developers check compliance with our coding standards. They use formal sign-off checklists, created by the Architecture team. Each review includes:

- The developer
- The coding supervisor, who is a more skilled developer that has been appointed by the Architecture team to conduct technical reviews
- Potentially other developers, who offer a peer perspective and who learn from the review process

No code review = no earned value for completion.

These are informal reviews, as defined by Section 3C.8 Technical Reviews.

These reviews, while necessary, are not by themselves sufficient. Quality also needs to be built in, *up front.* We do this by conducting "Writing High Quality Code" workshops, with mandatory attendance by developers. This half-day, interactive workshop includes:

We not only *inspect* quality, but also *expect* it.

- Presentation and discussion of good coding practices, with examples. Practices include, but are not limited to: decomposing your code, extracting variability, choosing between public and private methods, following standards, returning from methods at a single point.
- A group exercise to code review and improve a sample module.
- Two-person team exercises to code review a sample module. The workshop instructor gives personalized feedback afterwards.

The workshop is much more than just a rollout of the coding standards. It teaches and reinforces proven coding practices. Our experience has shown that such up-front training reduces time spent later in code reviews.

3C.12.7.4.2.3 Update Designs / Reverse Engineer Code

The developer's software modules have now passed code review. As we progress the software design through detailed design and into coding, we occasionally make coding decisions that affect the original software design. For example, during coding we may re-factor a key component interface into two smaller interfaces. The change to this key component interface should be reflected in the UML component model, which is a software design work product.

When we update or reverse engineer into the component model, we:

- Check for any impact on component interfaces[46] by using our traceability links (see below). This would imply an impact on upstream design documentation such as use cases, leading ultimately to requirements;
- Follow accepted change control processes before we update an already accepted design deliverable;
- Version manage the changed designs while we version manage the updated code

3C.12.7.4.2.4 Traceability, Traceability, Traceability

Our system better implements requirements because the design model translates directly into code modules. Our system better handles changes to requirements because our Java code is traceable back to the design. We deliver documented procedures that explain how to follow traceability links from code to designs, and ultimately to requirements.

The following paragraphs describe additional techniques that improve traceability during coding. Lastly, Section 3C.12.9 Test Management describes traceability from testing back to coding.

3C.12.7.4.2.4.1 Storing Detailed Design Information Directly in Java Classes

UML class diagrams are very good at specifying a class' name, methods, fields, and associations. But they are less useful for storing formatted, easily readable business descriptions of the class. The "Fast Track" approach makes innovative use of JavaDoc for these textual descriptions. Rather than storing the information in an external document, we store it directly in the class' source code. JavaDoc is a comment extraction tool. First, coders embed comments in the code, with HTML formatting instructions. Then, with a single command, JavaDoc extracts those comments into HTML pages. Naturally, this hyperlinked documentation is readable in a browser. Reviewers can easily move from one class description to another or dive into more detail.

Using JavaDoc properly, however, is more than just following the special formatting instructions. It involves writing code comments with the appropriate audience in mind. Many developers are taught to write very technical code comments. For example:

```
/* Call data handler to access case file and get cases
/* for NCP; use where clause of Support? as yes.  Check
/* returned set length parameter for 0.  Populate list
/* by instantiating iterator over non-NULL hash map -- but
/* remember to call constructor with zero-OK flag and
/* catch collection_create exception, …
```

[46] These are Application Programming Interfaces, as opposed to User Interfaces, as discussed elsewhere in the TMA when we talk about "interfaces" in the context of "components."

But our approach to self-documenting code makes additional, technical commentary like this is generally unnecessary. Moreover, these comments are cryptic – meaningful only to a technical audience. Instead, alliance coders write comments that are more business-oriented. As in:

We use JavaDoc for meaningful design commentary, stored directly in the code.

```
/* Do a quick existence check to determine whether the
NCP is
/* attached to any cases with support-ordered amounts.
/* If not, don't show on list to avoid having the user
/* double-click and display a blank page.
```

The JavaDoc utility extracts these comments into HTML. Using JavaDoc in this manner,

- Coders and less technical reviewers alike can discuss this documentation.
- Maintenance developers are much more likely to update the higher-level, textual documentation, because it is embedded directly in the code.

Our use of JavaDoc avoids shelf ware-bound documentation. The "Fast Track" approach is introduced in Section 3C.12.6 Software Detailed Design.

3C.12.7.4.2.4.2 Minimizing Hand-Offs

The alliance minimizes hand-offs from detailed design to coding, thereby supporting traceability from a "human perspective."

Some object-oriented projects separate their detailed design team from their coding factory. We believe that results in miscommunication and misunderstandings – ultimately, in code that does not properly meet requirements. We reduce hand-offs, such that one developer or team typically owns a class. We usually involve the developer up-front in the detailed design. This element of the "Fast Track" detailed design process is described in Section 3C.12.5 Software Detailed Design.

3C.12.7.4.2.4.3 Managing Change while Coding

Coding activities reveal design defects, requirement inconsistencies, and plan changes. Our coders exercise discipline, following prescribed change management procedures. See Section 3A.4 Change Request Management. As well, our documented procedures for following traceability links facilitate thorough impact analysis.

We monitor impacts of changes to usability, trainability, interface conformance, conformance with specified requirements, producibility, and supportability, as specified by IEEE. More generally, we control changes to the software, project and engineering plans, approved specifications, and established baselines. This addresses, respectively, the IEEE requirements in clauses 5.4.2 – 5.4.5 that relate to coding. We also update the project plan to reflect the latest thinking on the implementation activities.

3C.12.7.5 Proven Tools

3C.12.7.5.1 IBM WSAD a First Choice among Developers

IBM brings to the alliance its own WebSphere Studio Application Developer (WSAD). Ovum recently recognized IBM VisualAge for Java, which is bundled with WSAD, as a market-leading tool.[47] The following paragraphs describe this Integrated Development Environment (IDE) for Internet and component development.

3C.12.7.5.1.1 Fully Integrated IDE

WSAD is not just a Java editor, compiler, and debugger bundled under the same logo. Consider compilation. In other mainstream Java IDEs, the coder explicitly invokes a separate step for compilation. But with WSAD, as developers write and save individual methods, the compiler automatically compiles them in the background. As well, the compiler's auto-suggest feature recommends possible fixes for syntax errors. This allows the coder to immediately detect a compilation error and make the fix, while still thinking about the method just saved.

WSAD has built-in options for automatically formatting code. This contrasts with environments that require a special plug-in or separate utility.

WSAD also includes an HTML editor that supports "WYSIWYG." "What You See Is What You Get" means that HTML layouts look the same in the development environment as they do in the production environment.

WSAD integrates with leading source code management tools, like Merant PVCS or Rational ClearCase. This allows developers to check-out and check-in code from within the development environment. In contrast, a traditional approach forces developers to use a separate tool, check out the code files, and then load them into the development environment.

WSAD integrates with version control tools, object modeling tools, and IBM's own application server.

WSAD also integrates with Rational Rose, which we use for object modeling. This integration supports round-trip engineering from detailed design diagrams to Java code. This achieves traceability in both directions.

Lastly, IBM's WSAD integrates seamlessly with IBM's WebSphere, which we use for our application server. This integration means that Java components can be easily deployed to the application server for testing purposes. WebSphere provides an execution environment for Java code, as CICS provides an execution environment for COBOL modules. Section 4A.4 CSE System Environments discusses WebSphere in more detail.

[47] See *Ovum Evaluates: Web Development Tools,* April 2001

3C.12.7.5.1.2 Powerful Object Browsing

WSAD provides a very powerful object browser for managing and viewing Java classes. Ultimately, Java source code is stored in ASCII text files. In other Java IDEs, coders are exposed directly to these files, to the raw format. Rather than focusing on classes or methods, they focus on files. If they want to work on a class, for example, they need to know the particular file name and its directory location.

WSAD uses a different and unique approach. WSAD's object browser provides a layer on top of Java files. It provides several different views for organizing classes. WSAD programmers need not know the physical file name or location; they focus on the more important classes and methods.

Using an object browser allows developers to:

- Quickly find a class with a particular name
- Easily examine an inheritance hierarchy (i.e., a class and its superclasses), without having to load the separate Java files
- View a list of different implementations of the same-named method
- View a list of different senders of a particular method

WSAD users manage classes and methods, not files containing classes and methods.

Without this feature, developers would typically need to explicitly load the text files and use weaker grep-type searches.[48] These browsing capabilities, which support wildcard searching, allow developers to wander through the code base, just as Internet users wander through hyperlinked HTML.

Viewing different implementers and senders is valuable for traceability and impact analysis. WSAD is the dominant leader when it comes to object browsing and "where-used" reporting features.

3C.12.7.5.1.3 On-the-fly Compilation

Coders *regularly* want to test small snippets of Java. Normally this requires writing and compiling a small but complete test program, as in:

1. Writing the snippets as complete methods or classes
2. Creating main driver routines or harnesses
3. Compiling the code
4. Exporting the code from the environment
5. Figuring out a way to pass arguments into the program
6. Executing the program and inspecting the results outside of the IDE

[48] Grep is a well-known Unix utility for performing string searches in files. This capability is often referred to generically as grepping.

7. Jumping back into the IDE to make changes and repeating steps 3 – 7

In other words, a lot of overhead for a small piece of code. Normally developers avoid small-grained tests and wait until the class was complete and ready for testing in a legitimate test harness.

Clearly, this is not the most effective way to code. Immediate feedback, while working on a particular snippet, is superior to delayed testing. It costs less to fix problems the sooner they are detected.

WSAD avoids these machinations. It allows developers to:

1. Write lines of Java code, without having to include them as part of a full method or full class
2. Compile them on-the-fly inside the IDE without any harnesses or main routines
3. Immediately inspect the results inside the IDE

Moreover, with its powerful inspector feature, WSAD allows developers to continue playing with the results, by changing data and calling other methods on that data. No test program is required.

WSAD is the only mainstream Java environment to provide the incredibly powerful workspace and inspector features.[49]

3C.12.7.5.2 Handling Multiple Classes of Errors

The following paragraphs describe how we address the requirement to handle multiple classes of errors. We use WSAD's debugger supplemented by GRNDS' logging framework.

3C.12.7.5.2.1 WSAD's Powerful Debugger

These features allow developers to more easily trap and fix bugs.

WSAD's debugger automatically detects, diagnoses, and reports multiple classes of errors. Developers can set debugging breakpoints that are stored across debugging sessions. Developers can also set conditional breakpoints, such that debugging breaks only if a certain expression is true (e.g., "break if payment_date – date_due == x or if ...."). Lastly, WSAD supports multi-threaded debugging, for monitoring different threads that execute simultaneously. This supports detection of problematic interactions between threads in thin-client systems like CCSAS CSE. In a thin client system, multiple users run shared Java modules on the same server, at the same time.

[49] They are unique relative to other Java environments. IDEs for Smalltalk, another object-oriented language, have had these capabilities for years. Smalltalk coders will vouch for the productivity gains.

Granted, these features are fairly standard across Java debuggers. What distinguishes WSAD is its powerful, *interpreted* debugging functionality. At any break point, a developer can:

1. Edit the method that is currently beginning debugged
2. Recompile the method (by simply saving it, as mentioned above)
3. Re-start debugging execution at the beginning of the method, without totally starting over

Thus, the debugger, editor, and compiler are integrated *while* debugging. Developers can tinker with potential fixes and immediately test the results without exiting the debugger and starting over. This feature dramatically improves developer productivity.

WSAD also has special facilities for debugging and unit testing EJBs[50] within the application server. Typically, a developer follows a multi-step process to first deploy the EJB to the application server, before starting a debugging session. WSAD provides short-cut tools to skip many of these steps and begin debugging immediately.

WSAD hides the complexities of debugging an n-tier application.

Debugging in a distributed environment is very complex without the right tools. Some debuggers cannot track the flow of execution as it moves from client to server or from component to component. In such cases, a frustrated developer can only "step over" the component without stepping though the code inside. This is clearly ineffective for diagnosing code problems. WSAD, however, provides end-to-end debugging, across client/server as well as component boundaries.[51]

3C.12.7.5.2.2 Logging Framework

The WSAD debugger is our primary tool for detecting and diagnosing errors. In addition, our GRNDS architecture includes a logging framework. Coders use this framework to sprinkle logging statements throughout their code. This framework is configurable, as it can write statements to a file, database, WebSphere log, or even a screen terminal. Logging levels can be changed from "verbose / debug," which prints everything for debugging purposes, to "production only."

Our GRNDS logging framework also supports debugging.

Logging is an additional mechanism for detecting complex errors, as it presents a documented trace of the flow over time. Logs provide a big picture view of a particular thread of execution, which can be reviewed with coding supervisors.

3C.12.7.5.3 Code Quality Tool Expedites Reviews

[50] Enterprise JavaBeans, which are part of the J2EE specification
[51] This is true when using WSAD with IBM's WebSphere application server, which we plan to use. See Section 4A.4 CSE System Environments.

We use thorough coding standards and formal code reviews to manage quality. To automate part of the review process, we use Parasoft JTest as a code quality tool.

JTest includes dozens of pre-built rules. It detects outright errors, like the nefarious but ubiquitous "use comparison rather than assignment in an if statement." It also spots style problems that affect maintainability, like "avoid unused variables" or "avoid hard-coded constants."

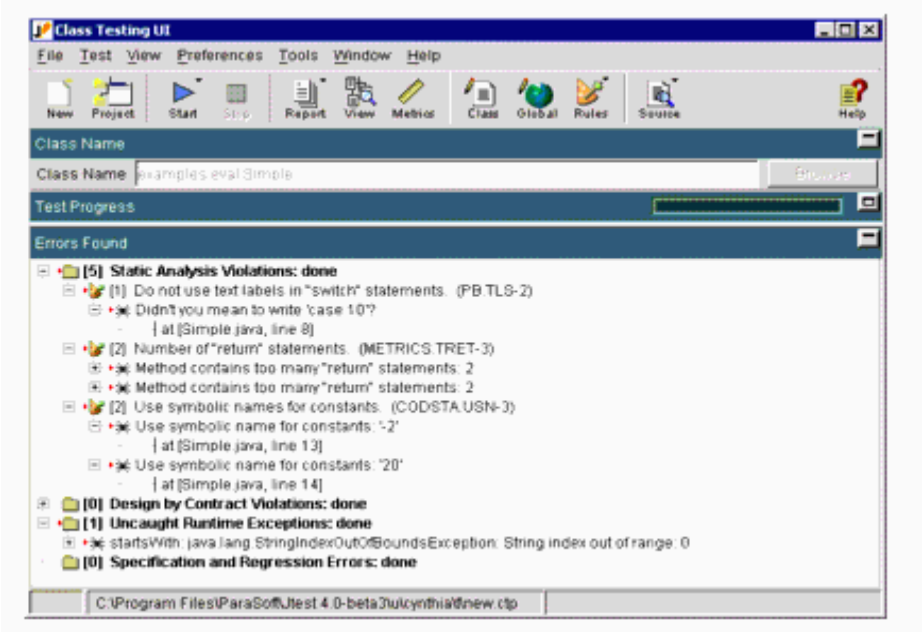


Figure 3C.12.7-4 Code Analyzer - *Automatic detection of coding violations will reduce time spent in code reviews.*

We can also add rules, based on our coding standards, through a graphical user interface.

JTest also reports on static code metrics. It scans code and produces useful reports on the number of classes, interfaces, public methods, and private methods; the average number of class variables per class; and other important metrics. The tools highlight problematic classes that exceed bounds provided by the tool or set by us. A class may be problematic because it is either "too intelligent", meaning it is too big with too many methods or variables, or "not intelligent enough", meaning it is too small to perform a useful, reusable function.

A code analyzer highlights bad style.

Metrics are particularly useful in an object-oriented (OO) environment. OO's focus on small-grained encapsulation results in projects containing hundreds of classes and thousands of methods. Metrics help us better understand the size and shape of our code.

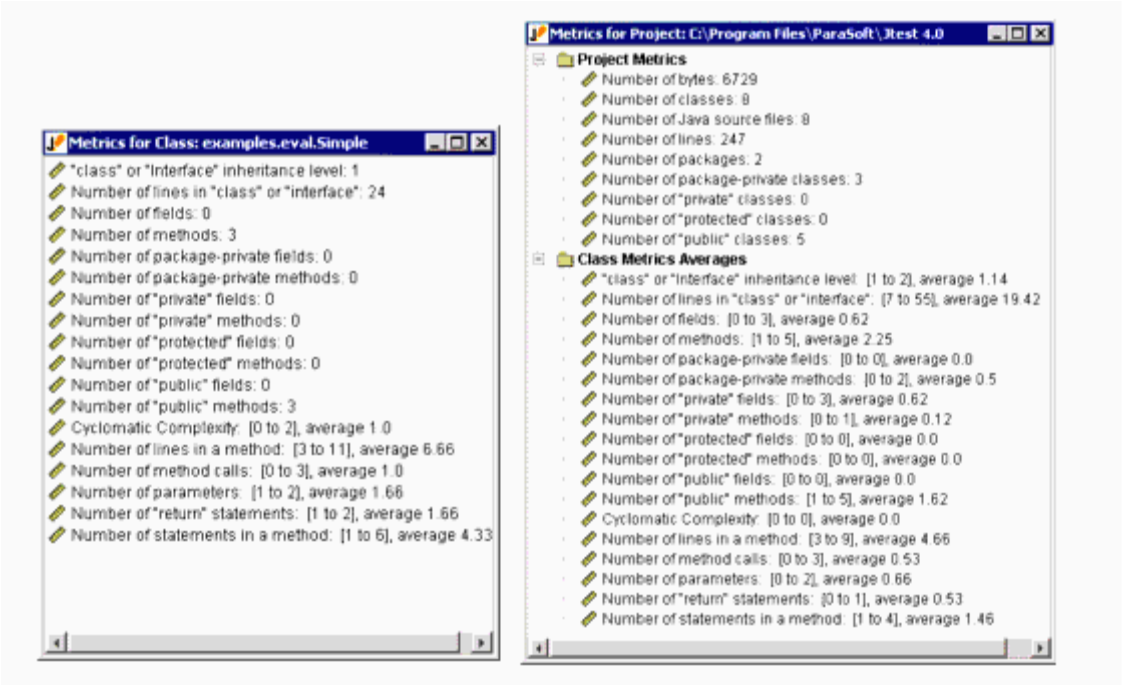


Figure 3C.12.7-5 Coding Metrics - *Metrics help us detect classes that are too intelligent – that have too many methods or fields.*

3C.12.7.6 Conclusion

Maintainability involves much more than taking a "Java in 21 Days" class, buying a Java IDE, and following the J2EE specification. It requires proven approaches, standards, and techniques. Our professional developers code in a disciplined way, combining leading-edge thinking with structured engineering principles.

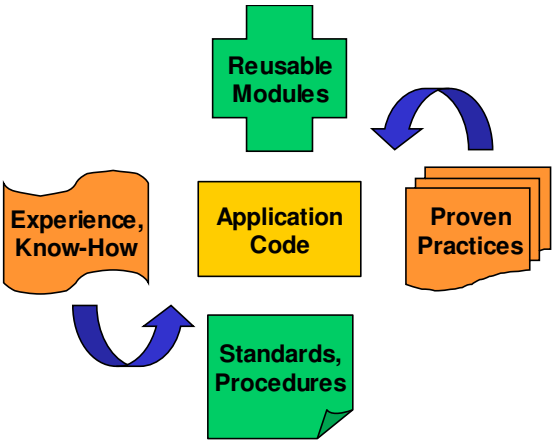


Figure 3C.12.7-6 Coding Toolkit - *Our coding professionals carry a comprehensive coding toolbox.*

Many large organizations have methodologies for structuring development activities. Our advantage is significant experience applying such approaches with newer technologies, on a large scale. Over and over and over again we have successfully delivered component-based systems of similar scope and size. We leverage the lessons learned on those projects for CCSAS. As well, we use a reference application, rigorous processes, and formal project management to guide development in a controlled fashion. Our forward-thinking approach to structured OO development leads to predictable delivery – predictable delivery of high-quality, maintainable code.

3C.12.8 System Integration

Some people think that getting heterogeneous systems to work together is as easy as buying and installing an eAI application. The alliance knows that system integration requires the same quality approaches that system building requires. The alliance brings an approach that provides highly configurable, adaptable, reliable, and manageable interface capabilities to deliver accurate and timely information to those who need it.

3C.12.8.1 Introduction

Interfaces are a key part of the CCSAS CSE application.

Workers rely on interfaces to send and receive information to locate non-custodial parents, offset tax refunds, and perform other critical activities – all focused on getting support payments in the door and out to those in custody of California's children. Moreover, interfaces to the State Disbursement Unit (SDU) help better automate payment processing. Therefore, it is vital that interfaces with the SDU and external agencies exchange information in a reliable, accurate, and timely manner. Otherwise, customer service and worker effectiveness of both the child support program and the interfacing agency will suffer.

Most child support systems in other states have used an approach based on custom-built, "point-to-point" interfaces. In this approach, each module knows the specifics about other modules with which it communicates. That is, the module hard-codes this information into its own application logic. The information includes not only the message format, but also the logic for routing, transforming (e.g., adding or removing fields), translating, and validating the message. It includes technical logic as well, perhaps for communications protocols, routing locations (e.g., FTP addresses), and secure transmissions. Bundling this logic directly into application modules complicates maintenance.

We instead propose an integration architecture based on:

- The **architecture layering principles** mentioned in Section 3C.12.2 System Architecture Design. These principles separate out the aforementioned logic from applications into a separate messaging layer. This simplifies maintenance.

- **eAI (Enterprise Application Integration)**. eAI is the next generation of technology and approaches for external integration. It is a set of technology services and a common framework for sharing data and workflows across disparate systems. eAI applies to integration both inside (e.g., Siebel and custom CSE components) and outside (e.g., Locate interfaces) the organization. This strategic enabler will help DCSS transform the way its systems interact with each other and with external agencies and companies. Using eAI untangles the mess of point-to-point interfaces, resulting in streamlined and more robust integration software.

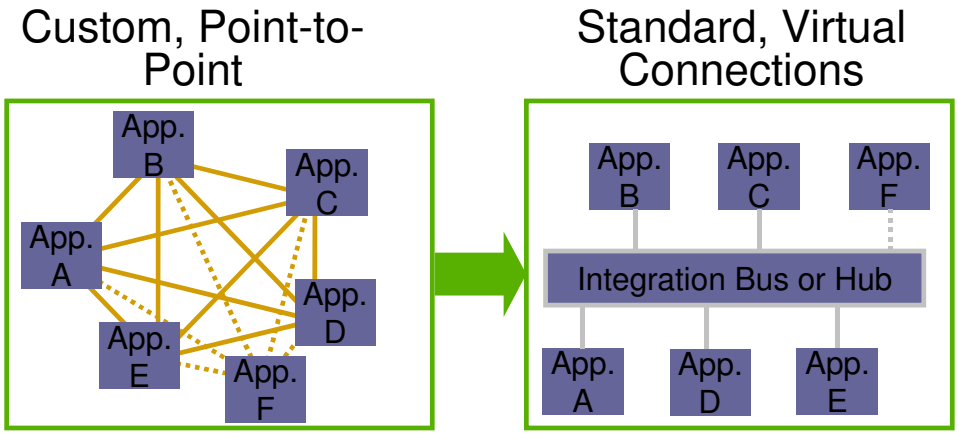


Figure 3C.12.8-1 Using a Proven Integration Approach - *Our solution incorporates the use of an integration bus to exchange information via common, reusable methods. An integration bus is a common, enterprise-wide way to communicate between applications.*

We crafted our system integration approach based on a thorough study of the Business Requirements List; our understanding of how California consortium systems interface with agencies today; and our experience designing, developing, and managing the interfaces used by other state Child Support Enforcement applications. This section proposes our proven principles, proven work products, proven process, and proven tools for system integration. Our architecture enables child support applications to exchange information via common, reusable methods and infrastructure. We understand the methodology and architecture for maintainable and high-performance integration. **We understand how important it is to get it right with the SDU.**

The alliance brings significant experience and assets for a jump-start on system integration.

As well, integration goes beyond just interfaces. This section addresses the integration and testing of hardware and software, to meet the State's requirements.

3C.12.8.2 Total System Integration Responsibilities

3C.12.8.2.1 Roles and Responsibilities

The development team builds the interface functions specific to the CSE application for exchanging data with the SDU and external entities. The alliance works with the State, SDU vendor, and external entities throughout the development and implementation process. The alliance expects the State to be involved throughout the system integration effort. The State provides a deep understanding of California's unique requirements. Moreover, the State's relationship with the SDU vendor and external entities is critical to successfully completing CCSAS system integration activities. The following table describes the general responsibilities of the alliance, the SDU vendor, the State, and external entities throughout the development and implementation. The final responsibilities may ultimately differ from those represented in this table.

Responsibilities	alliance	SDU	State	External Entities
Defining requirements, designing, building, and testing application functions internal to the SDU application		●	●	
Defining requirements for SDU/CSE interface functions residing in the SDU environment	●	●	●	
Designing, building, unit testing, and unit integration testing SDU/CSE interface functions residing in the SDU environment		●	●	
System testing[52] SDU/CSE interface functions residing in the SDU environment	●	●	●	
Defining requirements, designing, building, and testing application functions internal to the CSE application	●		●	
Defining requirements, designing, building, unit testing, and unit integration testing SDU/CSE Interface functions residing in the CCSAS environment	●	●	●	
System testing SDU/CSE interface functions residing in the CCSAS Environment	●	●	●	
Defining requirements for External Entity/CSE interface functions	●	●	●	●
Designing, building, unit testing, and unit integration testing External Entity/CSE interface functions residing in the CSE environment	●		●	
System testing External Entity/CSE interface functions residing in the CSE environment	●		●	●
Designing, building, unit testing, and unit integration testing External Entity/CSE interface functions residing in the External Entity environment				●
System testing External Entity/CSE interface functions residing in the External Entity environment	●		●	●

[52] In this section, "system testing" includes the informal system test and the formal System Verification Test.

Responsibilities	alliance	SDU	State	External Entities
Defining and allocating performance requirements between the SDU and CSE systems	●	●	●	
Managing integration risk	●	●	●	●
Scheduling and conducting SDU/system integration status meetings	●	●	●	
Participating in SDU/system integration status meetings	●	●	●	
Developing SDU/integration work plan and schedule	●	●	●	

Figure 3C.12.8-2 System Integration Responsibilities. *The alliance works with the State, SDU, and external entities throughout the development and implementation process.*

The alliance's Chief of Development, Requirements/Design Manager, and Application Development Manager own the following alliance responsibilities:

- Defining requirements for the SDU/CSE interface functions residing in the SDU environment
- Defining requirements, designing, building, and testing application functions internal to the CSE application
- Defining requirements, designing, building, unit testing, and unit integration testing the SDU/CSE Interface functions residing in the CSE environment
- Defining requirements for External Entity/CSE interface functions
- Designing, building, unit testing, and unit integration testing External Entity/CSE interface functions residing in the CSE environment
- Defining and allocating performance requirements between the SDU and CSE systems
- Managing integration risk
- Scheduling and conducting the SDU/CSE system integration status meetings through completion of the design phase
- Developing the SDU/CSE integration work plan and schedule

As part of these activities, the Chief of Development:

- Oversees and supports the development team in performing system integration activities
- Conducts joint integration planning sessions with the SDU vendor as necessary
- Approves the jointly developed work plan and schedule for SDU integration
- Reviews the status of CSE application system integration activities
- Participates in joint issue resolution meetings with the SDU vendor, external entities, and the State as necessary

As part of these activities, the Requirements/Design Manager:

- Participates in joint integration planning sessions with the SDU vendor

- Works with the SDU vendor to jointly develop an integrated work plan and schedule for SDU integration
- Monitors and reports status of requirements/design system integration activities with the SDU and external entities
- Identifies and manages system integration risks
- Conducts joint status meetings with the SDU vendor during the planning, requirements, and design stages of the project
- Leads the requirements team in analyzing and documenting SDU and external interface requirements
- Leads the requirements/design team in designing a solution based on the documented requirements

As part of these activities, the Application Development Manager:

- Participates in joint integration planning sessions with the SDU vendor
- Monitors and reports status of SDU and external entity interface development activities
- Identifies and manages system integration risks
- Conducts joint status meetings with the SDU vendor
- Identifies and manages system integration risks
- Conducts joint status meetings with the SDU vendor during the application build and unit test portion of the project
- Leads the development team in coding, unit testing, and unit integration testing CSE interface functions with the SDU and external entities
- Participates in impromptu test defect resolution meetings as necessary

The alliance's Chief of Architecture and Technical Manager manage the following alliance responsibilities:

- Defining requirements for the SDU/CSE interface functions residing in the SDU environment
- Defining requirements and assisting the development team with designing, building, and testing application functions internal to the CSE application
- Defining requirements and assisting the development team with designing, building, unit testing, and unit integration testing the SDU/CSE Interface functions residing in the CSE environment
- Defining requirements for External Entity/CSE interface functions
- Defining and allocating performance requirements between the SDU and CSE systems
- Supporting the development team in designing, building, unit testing, and unit integration testing External Entity/CCSAS interface functions residing in the CSE environment
- Supporting the testing team in system testing the SDU/CSE interface functions residing in the CSE, SDU, and External Entity environments

As part of these activities, the Chief of Architecture:

- Oversees and supports the architecture team in performing system integration activities
- Participates in joint integration planning sessions with the SDU vendor as necessary
- Participates in joint issue resolution meetings with the SDU vendor, external entities, and the State as necessary
- Works with the SDU vendor and State to allocate SDU interface performance requirements between the SDU and CSE applications

As part of these activities, the Technical Manager:

- Participates in joint issue resolution meetings with the SDU vendor, external entities, and the State as necessary
- Oversees architecture team support of CSE application interface design and development efforts
- Oversees architecture team support of CSE system test efforts

The alliance's Chief of Testing manages the following alliance responsibilities:

- System testing the SDU/CSE interface functions residing in the SDU environment
- System testing the SDU/CSE interface functions residing in the CSE environment
- System testing External Entity/CSE interface functions residing in the External Entity environment
- Identifying and managing system integration risks
- Conducting joint status meetings with the SDU vendor during the system test phase of the project

As part of these activities, the Chief of Testing:

- Coordinates system test activities with the application development and architecture teams
- Oversees and supports the system testing being planned and executed by the testing team
- Participates in joint issue resolution meetings with the SDU vendor, external entities, and the State as necessary
- Coordinates system test activities with the SDU and external entities

3C.12.8.2.2 Risks and Mitigation Strategies

The alliance uses a systematic risk management approach. We identify and assess risks, determine cost-effective risk reduction actions, and monitor and report progress. On an ongoing basis, we work cooperatively with SDU vendor staff, external entity staff, and CCSAS project staff to identify, track, and resolve system integration risks and issues. We follow our overall project risk management approach when dealing with

system integration risk. The remainder of this section describes the following risks and corresponding mitigation strategies:

- Quantity and diversity of external entity systems
- Conflicting implementation schedules
- Collaboration with interfacing entities
- Changes or additions to interface requirements due to external entities or SDU vendor initiatives

3C.12.8.2.2.1 Quantity and Diversity of External Entity Systems

The alliance mitigates risks associated with the quantity and diversity of external interfaces by using an architected approach, by leveraging eAI technologies, and promoting standards and guidelines that facilitate flexibility and adaptability.

eAI provides a common messaging framework for integrating data and end-to-end business processes across disparate applications. eAI is especially built to manage complexity across many different systems, written in different languages, communicating in different protocols, and using different message formats.

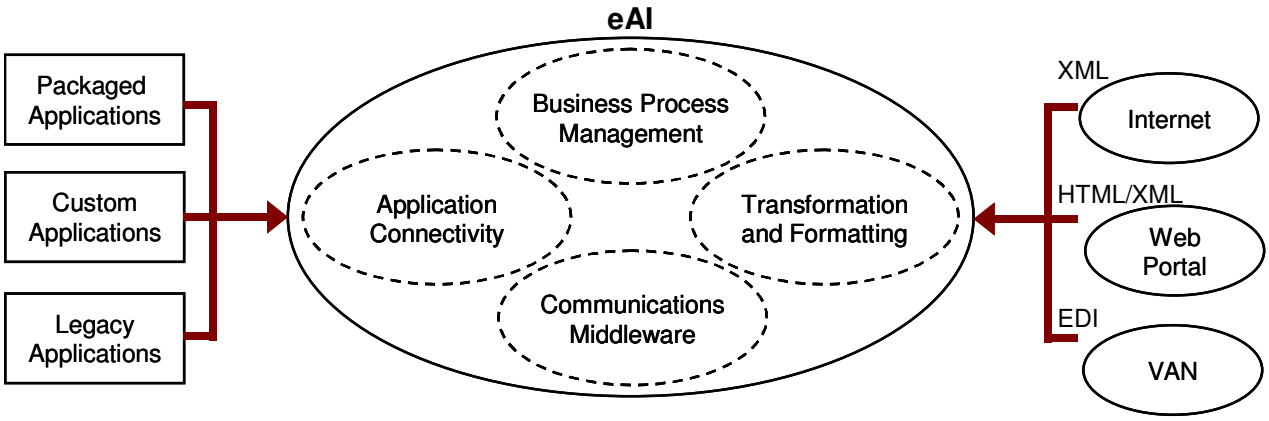


Figure 3C.12.8-3 Enterprise Application Integration - *Our solution incorporates an eAI bus to support CCSAS CSE application integration needs today and adapt to future changes.*

For example, eAI allows us to use XML as a common message format (explained elsewhere in this document). Where other agencies use different formats, like comma-delimited or fixed-length, eAI provides converters and tools for us to easily transform to and from XML.

eAI provides four major functions:

1. **Communications middleware** – provides network-based locations (queues) where applications can place data for the purpose of communicating with other processes

2. **Application connectivity** – provides logic on how and when to communicate with applications outside of the eAI space
3. **Transformation and formatting** – provides logic on how data should be validated, formatted, and transformed
4. **Business process management** – provides logic to support communication with other enterprises using standard communications protocols and data formats (e.g., EDI, XML, SOAP)

The eAI bus provides the CSE application with a means to exchange information across disparate systems, without rebuilding the external entities' legacy interfaces.

3C.12.8.2.2.2 Conflicting Implementation Schedules

The external entities and the SDU vendor may have different priorities and goals, which may cause schedule conflicts. Because the CSE application and SDU depend on each other, coordination between these two is vital.

The alliance mitigates the risk of implementation schedule conflicts. We work closely with the selected SDU vendor to identify:

- Dependencies
- Points of intersection in our respective schedules
- Timing of reciprocal participation in joint system integration activities, discussed below

In this manner, we interact with the SDU vendor and develop a mutually agreeable schedule and integrated plan.

Moreover, we account for reasonable change in the CSE application's SDU integration approach. Specifically, we tailored our approach to remove the dependency on having the SDU operational early in the project.

3C.12.8.2.2.3 Collaboration with Interfacing Entities

We collaborate with external entities in developing interfaces. This avoids schedule delays resulting from rework, interface maintainability problems due to lack of standardization, and interface data integrity problems.

Our mitigation plan includes early and frequent communication with external entities. The alliance conducts joint system integration activities to foster coordination and collaboration.

3C.12.8.2.2.4 Changes or Additions to Interface Requirements

The alliance mitigates risks associated with changes to CSE interfaces by:

- Leveraging the capabilities of the eAI bus (as mentioned earlier in this document)
- Using XML as the internal format, to build flexible, adaptable, and maintainable interfaces

The eAI bus's data transformation services allow us to change the format or communication protocols we use for sending data to an external entity – without code changes. This becomes increasingly important as external entities replace their systems and adopt a universal format such as XML. These data transformation capabilities also support the addition of new interfaces that require the same data as an existing interface, but in a different format.

As well, our Interface Design Descriptions (IDDs) clearly define data exchanges. This mitigates the risks associated with changes or additions to CSE application interfaces. IDDs clearly communicate to internal and external entities how the interchange occurs, including information on sender, receiver, message formats, frequency, and parameter types and lengths.

3C.12.8.3 Approach to Coordinating CSE System and SDU Schedules

3C.12.8.3.1 Communication

Ongoing communication with the external agency and SDU vendor staff is crucial. Other government agencies – and even programs in the same agency – often have different and changing priorities. This can delay system integration activities. As such, our approach focuses on early and frequent communication with those responsible for interfacing systems. In particular, the alliance uses a combination of formal and informal communication activities to facilitate system integration. This fosters a collaborative environment for coordinating system integration plans, schedules, designs, tests, and maintenance activities.

We conduct formally planned system integration activities with the SDU vendor and other interfacing organizations throughout interface development. We:

- Conduct joint status meetings
- Produce status reports
- Hold joint requirements sessions
- Hold joint design meetings periodically

We use the alliance project office to conduct joint meetings and sessions.

In addition to formal planned activities, we hold impromptu meetings between the alliance and integrating system staff to review test results and discrepancies, discuss

change requests, and address performance issues. We similarly host these meetings at the alliance project office.

These formal and informal systems integration activities foster communication and support the development of system interfaces. However, one or more of these activities may not be feasible or necessary for certain interfaces. For example, staff supporting the Credit Bureau's system may not be able to participate in joint meetings. Moreover, these meetings may be unnecessary, because the Credit Bureau publishes a standard data exchange format. In those situations, the alliance instead relies on impromptu and informal communications with interfacing system staff.

We believe this communication strategy provides the foundation for a collaborative environment, for successfully developing interfaces with the SDU vendor and external entities. The communication activities with the SDU vendor throughout development are described in Section 3C.12.8.3.2 Joint System Integration Activities and summarized in Figure 3C.12.8-4 below.

SDU Interface Lifecycle Phase	Joint Status Meetings	Joint Requirements Sessions	Joint Design Sessions	Impromptu Meetings & Informal Communication
Planning	●			●
Software Requirements	●	●		●
Software Design	●		●	●
Software Coding and Unit Test	●			●
Unit Integration Test	●			●
System Testing	●			●
System Qualification Testing	●			●
Pilot and Implementation	●	●	●	●
Application Maintenance		●	●	●

Figure 3C.12.8-4 SDU Interface Communication Activities – *The alliance establishes and maintains communication with the SDU vendor throughout the project.*

3C.12.8.3.2 Joint System Integration Activities

Collaboration between the alliance and the SDU vendor is necessary to construct a reliable and accurate interface for child support collection and disbursement. As discussed earlier, the alliance builds a relationship with the SDU vendor by conducting formal activities and impromptu sessions throughout interface development. This section focuses on areas of reciprocal participation in the context of the SDU interfaces. We also conduct many of these activities with other external agencies.

3C.12.8.3.2.1 Planning and Scheduling

The alliance develops an SDU integration plan. This includes a description of the technical solution, integration, alliance contractor responsibilities and deliverables, audits and controls, and performance requirements that span the SDU and CSE applications. It also outlines how we allocate these requirements between the two systems.

The State has yet to select an SDU vendor, and the alliance may be responsible for delivering the integration plan prior to SDU vendor selection. Therefore, we may need to revise the SDU integration plan based on the SDU vendor's proposed solution. Monthly status meetings will be forums for reviewing the SDU integration plan with the SDU vendor and mutually agreeing to plan revisions. Such revisions are subject to State approval.

We conduct joint monthly status meetings with the SDU vendor during the development and implementation of the SDU interface. Initially, these meetings focus on coordinating design, development, testing, and implementation activities. Through these discussions, the alliance and SDU vendor identify dependencies, points of intersection in our respective schedules, and needs at those points of intersection. With this, we develop a mutually agreeable work plan and schedule for the CSE/SDU interface.

Ongoing status meetings between the SDU vendor and alliance focus on monitoring progress against the integrated work plan, addressing schedule variances, and resolving issues.

3C.12.8.3.2.2 Reviewing and Clarifying Requirements

The collections process is distributed across the SDU and the CSE application. Both systems must therefore work in concert for timely collection processing. The alliance conducts joint requirements review sessions with the SDU vendor. These sessions clarify the CSE interface requirements defined by the alliance and approved by the State during requirements analysis. Since the SDU vendor is responsible for developing interfaces that meet these requirements, reviewing the integration requirements with the SDU vendor early in interface development avoids rework in design, development, and testing.

These sessions also provide a forum for identifying additional requirements, clarifying the approach for integrating the two systems, and identifying constraints. By holding these joint sessions prior to design, the alliance and SDU vendor both avoid rework and delays that could otherwise result from misinterpreting or omitting requirements.

3C.12.8.3.2.3 Designing

Collaboration between the alliance and the SDU vendor continues into interface design. Rather than designing each system's portion of the interface in isolation and hoping the results will integrate, the alliance conducts joint design sessions. These sessions

leverage the expertise of each vendor regarding the collections process as well as their respective systems. This builds reliability, accuracy, and timeliness into the interface. During these sessions, the alliance and SDU vendor perform data mapping, resolve issues, clarify assumptions, and review each other's designs for compatibility.

Integrating two systems requires not only joint design sessions, but also the right resources. Integrating the SDU application and business processes with that of the CSE application requires both a technical and functional perspective. We bring the functional analysts and technical architects of both vendors together during joint interface design sessions.

The alliance uses XML documents in review sessions to discuss the contract, or interchange, between the two systems. This does not require that the SDU use XML. As mentioned, the eAI bus allows us to convert from XML into many other formats. We use XML in these meetings, though, to provide a human-readable and unambiguous view of the interface definition. XML has markup tags that explain field names in plain English. Our use of XML is explained elsewhere in this document.

3C.12.8.3.2.4 Testing

The alliance subjects CSE interface logic to the same rigorous testing as for other system components.

Integration adds a new requirement for joint testing, across different organizations. We jointly test interface functions to confirm that the interface functions as expected, prior to production use. This mitigates problems that would otherwise arise in production.

To assist the SDU vendor with testing, the alliance provides the SDU vendor with test files representative of the transactions that the CSE application communicates to the SDU. Likewise, we expect the SDU vendor to provide the alliance with test files to facilitate testing in the opposite direction.

The alliance works with the SDU vendor to develop an acceptance test plan and conduct user acceptance testing for the SDU/CSE interface. In addition, we may hold impromptu defect resolution meetings with the SDU vendor to discuss testing issues.

3C.12.8.3.2.5 Piloting

Our approach supports local payment receipting and local disbursement. As such, the CSE application pilot does not depend on the SDU. However, our approach does not eliminate the need to pilot SDU functionality and coordinate with the SDU. The alliance will define the approach to piloting the SDU and coordinate pilot activities with the SDU vendor. This pilot starts small and scales gradually by:

- Leveraging consortium application payment receipting capabilities. In this way, a relatively small portion of the payment receipting volume can be diverted to the SDU initially.

- Increasing the receipting volume at the SDU incrementally. This allows us to resolve operational issues, staffing needs, or application problems gracefully.

This pilot approach allows the State of California to avoid operational difficulties that other states have experienced. It allows operational practices and application functionality to be fine-tuned prior to full-scale operations.

The alliance expects to conduct meetings with the SDU vendor periodically during pilot planning and execution. These meetings provide a forum for identifying and resolving performance issues with collection processing. This includes those portions of the process performed at the SDU, CCSAS Statewide Services, and LCSAs.

Prior to SDU pilot execution, the alliance conducts joint planning sessions with the SDU vendor at the alliance project office. Through these sessions, the alliance and SDU vendor determine a mutually agreeable approach for incrementally diverting payments to the SDU. The alliance and SDU vendor also develop an integrated schedule/work plan for performing SDU/CSE pilot activities.

We hold weekly status meetings with the SDU vendor until pilot completion. These meetings focus on monitoring progress against the integrated work plan, addressing schedule variances, and resolving performance issues with the SDU/CSE interface. During pilot execution, we discuss SDU, CCSAS Statewide Services, and LCSA processing issues, and we communicate action plans for resolving these issues.

As during SDU/CSE interface development, we hold informal or impromptu joint design, testing, and defect resolution sessions with the SDU vendor during pilot execution, as necessary. To assist the SDU vendor with testing interface changes resulting from pilot operations, the alliance continues to furnish the SDU vendor with test files representing the transactions that the CSE application communicates to the SDU. Likewise, the SDU vendor provides the alliance with test files for testing in the opposite direction.

3C.12.8.3.2.6 Maintaining

The need to communicate and coordinate with the SDU vendor does not cease after the implementation of the SDU interface. Application maintenance and enhancement activities directly or indirectly impacting the SDU interface require coordination between the alliance and SDU vendor.

For example, the CSE application's SDU interface components might require modification post implementation. Then, the alliance would apply the communication techniques used during the initial development of the interface. This includes joint design sessions and impromptu defect resolution meetings, as appropriate.

3C.12.8.3.3 SDU Implementation Schedule

The alliance coordinates implementation schedules and activities with the SDU vendor as well as other external agencies. While this section addresses schedule coordination

in the context of the SDU, many aspects of this approach apply to other external agencies.

The State has not yet selected an SDU vendor. As such, the alliance provides flexibility and reduces dependencies on the implementation schedule of the SDU with a phased approach. The alliance uses a phased implementation of CSE application capabilities. In the interim, the alliance uses consortium system capabilities. This provides the State schedule flexibility in procuring an SDU solution.

As outlined in the Section 4.1 of the business solution, our SDU integration approach in Version 1 involves transferring collection data and funds to the LCSAs for distribution and disbursement. One SDU interface forwards information on collections receipted at the SDU to CCSAS Statewide Services. CCSAS Statewide Services uses this information to allocate the collection to the appropriate LCSAs. It forwards the payment data to these LCSAs for distribution and disbursement.

Another SDU interface supports the transfer of funds to the LCSAs. They then have the funds underlying the collection that they distribute and disburse. In essence, this second interface between the SDU and CCSAS Statewide Services involves sending the SDU disbursement instructions. These instructions tell the SDU how much should be disbursed to each LCSA or to a custodial parent in the case of Non-IVD collections.

In Version 1, LCSAs operate consortium systems linked to CCSAS Statewide Services for access to cross-county functionality and data. The LCSAs' consortium systems support the receipting, distribution, and disbursement of collections. Therefore, the SDU need not be established to implement CCSAS Statewide Services. In other words, even without the SDU, we can begin realizing the business benefits derived from the other capabilities in CCSAS Statewide Services for Version 1. The ability to continue processing collections at the LCSAs provides independence. The alliance can implement CCSAS Statewide Services without the SDU. This schedule flexibility allows for schedule variance with regard to SDU integration.

Moreover, our approach to SDU integration reduces the integration burden on the SDU vendor during Version 1. We do this by requiring only two interfaces and deferring a larger portion of the integration until Version 2. Thus, our approach provides schedule flexibility by reducing the scope of CCSAS integration work expected from the SDU vendor over the near term.

Our approach to testing CSE interfaces naturally involves exchanging test data with the SDU vendor. However, the alliance also uses stub programs where required to facilitate independent testing of CSE interfaces. These would be helpful, for example, were the SDU vendor unable to conduct joint testing activities according to a previously agreed upon schedule. This does not replace the need for jointly testing the interface. However, we can simulate the exchange of data and perform further validation prior to joint testing. This further reduces delays by resolving issues that would otherwise have been uncovered later, during joint integration testing.

Our approach reduces dependencies between the SDU vendor's implementation schedule and the alliance's schedule. But it does not eliminate the need for coordination. As discussed elsewhere in this document, the alliance establishes communication early with the SDU vendor, keeping lines of communication open throughout development.

3C.12.8.4 Managing CCSAS CSE Interfaces With External Entities and the SDU

3C.12.8.4.1 Proven Principles

The following paragraphs discuss general principles, applying to both external entities and the SDU. This is followed by a discussion of principles specific to the SDU.

It is worth noting that during the Version 1 system build process, the local systems in use by the LCSAs form an important integration point. Because the local LCSA systems are external to the CCSAS Version 1 software, the alliance will address this integration using the same processes as any other external entity (defining design in the External Entity Interface Design Description).

3C.12.8.4.1.1 General Integration Principles

The alliance brings extensive experience in performing thorough Systems Integration using eAI. Key principles that guide us in this analysis include:

1. Layer integration logic
2. Apply component principles
3. Build common architecture frameworks and services
4. Define common message structures
5. Apply accepted standards
6. Use well-formed metadata to define interfaces

The following paragraphs describe these principles.

3C.12.8.4.1.1.1 Layer Integration Logic

The alliance believes firmly in the concept of layering a system's architectural and business responsibilities. Proper layering of logic isolates changes and reduces the overall cost of maintenance. These layering principles are explained in the system and software architecture sections. They apply to the integration architecture as well.

eAI vendors sell tools which allow application developers to build business logic along side with integration logic. However, these integration tools cannot be relied upon to separate this logic correctly on their own. Instead, we perform careful architectural design to improve maintainability.

Integration logic is comprised of several different types of rules. Rules regarding routing, data translation, data enrichment (i.e., supplementing messages with additional information), and application communication need to be architected to allow for improved flexibility of design.

The alliance breaks these different types of integration rules into separate technical modules. This supports the overall principle of layering. Our proven tools support this separation by allowing us to put the different types of code in separate modules. The alliance understands that even with eAI tools, you can still write non-maintainable code that is difficult to understand. Our experience and proven developer's toolkit (discussed elsewhere in this document), along with our layering approach, avoids "spaghetti" eAI code.

For instance, we build application communication logic in adapters that are closest to the external application. This logic defines the technical handshaking that must be performed with the external application along with rules stating the location of data sources and when data should be obtained.

We build routing and data transformation rules separately, in integration modules distinct from the adapters. Technical services are accessed by calling common procedures using a well-defined Application Programming Interface (API).
The diagram below illustrates a generic implementation of these layering principles, which separate out different types of logic.

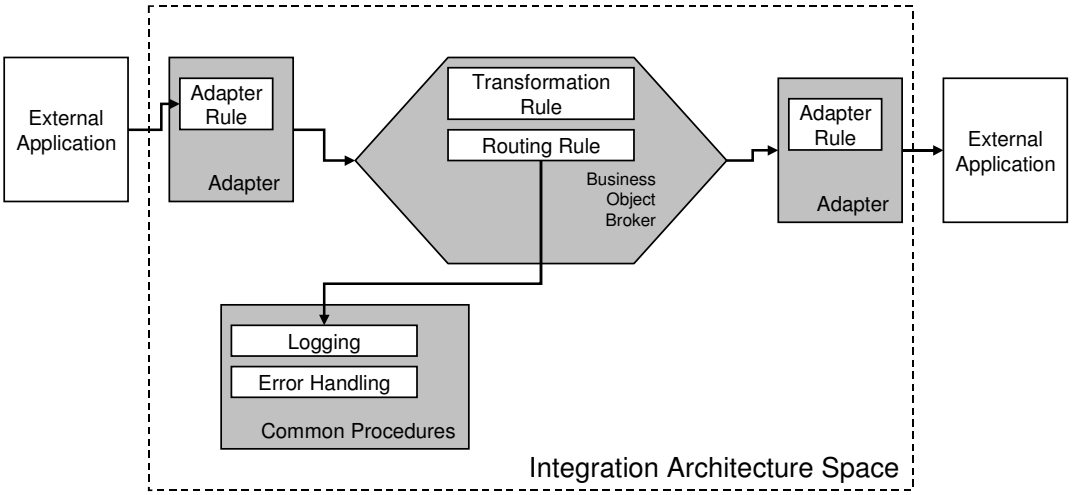


Figure 3C.12.8-5 Separation Of Integration Logic - *Our approach separates integration logic into multiple layers, easing maintenance and reducing the cost of change.*

3C.12.8.4.1.1.2 Apply Component Principles

Object-oriented and component design principles have been established practices for many years for decomposing large software systems. What differentiates the alliance is how it reuses these proven practices, from the software domain, in the newer space of

eAI. Along with fundamental architecture layering principles, we apply standard component principles to design the CCSAS CSE integration architecture. As defined in Section 3C.12.2 System Architecture Design, a "component" is an entity that performs a set of related business functions and can only be accessed through a set of well-defined, application programming interfaces.

CCSAS CSE integrates with numerous external agencies and businesses. This might involve hundreds of things to manage, like message formats, queues for temporarily storing messages, and modules for routing, transformation, validation, and aggregation. In a naïve eAI implementation, the system management console would list all of these items. A system architect might devise a clever naming scheme, with prefixes to help sort everything in the console. This might even appear to be logically organized.

Organized, that is, until maintenance needs to make changes both big and small. Keeping track of these different bits and pieces by naming standards alone becomes unwieldy for large systems. What CCSAS CSE needs, and what the alliance brings, are concepts of system architecture design to eAI. Most importantly, we apply principles of component encapsulation to decompose a large eAI system into smaller parts – eAI components. This approach results in a federation of loosely coupled eAI subsystems.

We also used well-defined message interfaces (again, a principle from component development) to hide the internal details of these eAI components. In this way, changes internal to an eAI component are isolated to that component. That is, they do not affect the rest of the integration architecture space.

The alliance has purposefully selected a market-leading eAI bus that supports this type of component grouping. SeeBeyond's e*Gate achieves this grouping, or encapsulation, by implementing multiple *schemas,* one for each eAI component. Schemas can communicate with each other but it must be done so using a special type of adapter and using an agreed upon message format. Each schema acts as a separate component and is managed separately. Each schema can be version controlled and upgraded separately. Most importantly, each schema can be extended and maintained separately.[53]

The diagram below illustrates a multiple schema integration space.

[53] As with components, being able to restrict changes inside the eAI component are only true as long as no changes are required to its "interface" – that is, the well-defined message formats the schema uses to integrate with other schemas.

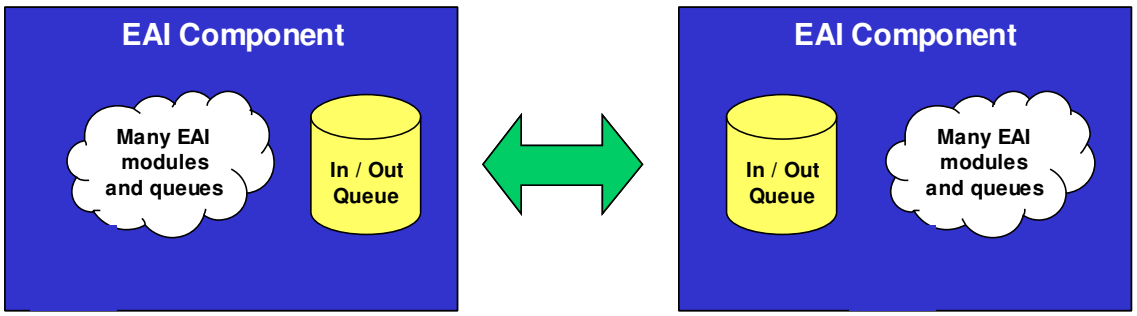


Figure 3C.12.8-6 Multiple Schema Approach. *Our approach collects like integration parts into schemas. Each schema is similar to a business component that can only be accessed via well-defined interfaces and that hides implementation complexities in the component.*

3C.12.8.4.1.1.3 Build Common Architecture Frameworks and Services

Similar functions are required by much of the logic in the integration architecture. These functions include error handing, audit logging, creation of global unique identifiers, and others. As can be expected, eAI vendors – including SeeBeyond – provide a certain level of these capabilities, out-of-the box.[54] However, our experience has shown that the vendor-provided capabilities can be implemented in ways that are less than optimal. Also, the vendor-provided capabilities usually do not meet all of our needs. The alliance separates these vendor APIs and new technical services into a different architectural layer. By doing so, we:

- Shield ourselves from future vendor API changes
- Hide unnecessary parameters from the developer
- Can automatically generate parameters in a consistent matter
- Give ourselves flexibility to change the implementation of these services, without affecting the integration code that calls them

The alliance architecture team develops and maintains these types of value-added services. The development of common services by a central team reduces cost and increases the consistency of the services. These services, in the eAI space, are analogous to the GRNDS frameworks of the J2EE space.[55]

Layering also eases the integration of these services with other areas of architecture, including the operations and development architecture.[56] For instance, a call to the error handling service might trigger some hook into the operation architecture monitoring services. By encapsulating error handling into a separate service, we hide technical complexity from the developer. The developer need only know how to call the error handling function.

[54] "Out of the box" is colloquial, meaning that when you buy a software package and take it out of the box, you automatically have a certain set of features.

[55] See Section 3C.12.4 Software Architecture.
[56] Section 3C.12.2 System Architecture defines "Operations" and "Development" Architecture.

Accenture brings to the alliance its SeeBeyond developer's toolkit. This contains proven assets for designing and implementing the eAI infrastructure. It includes:

- Tools to aid in the selection of vendor products
- Design aids
- Developer's handbooks
- Administration guides
- A common function library, including error handling and logging

In this way, the alliance has a jump-start with proven frameworks.

3C.12.8.4.1.1.4 Define Common Message Structures

The alliance defines common message structures, with the benefits of reuse and the entire enterprise in mind. This is analogous to defining an enterprise-wide business object model in a component-based environment.

For instance, suppose an interface is developed between external system "A" and external system "B" where a payment message is passed between them. Suppose later two other external systems ("C" and "D") also will pass a similar payment message. If these interfaces are not considered together, the resulting solution will appear as below. The structure definitions for Payment and Payment prime will differ.

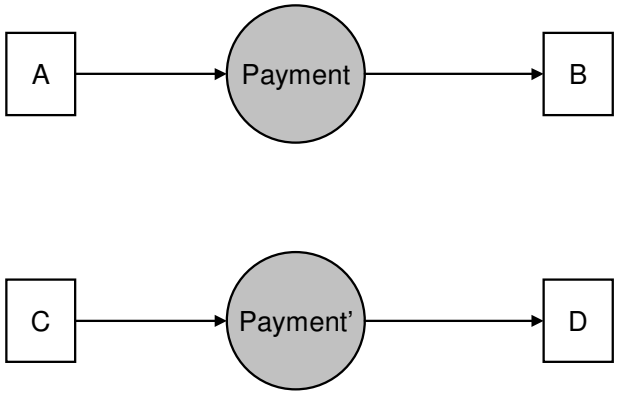


Figure 3C.12.8-7 Point-To-Point Approach - *Without a coordinated design of common message structures, point-to-point interfaces can exist even while using eAI technologies.*

This is similar to having two point-to-point interfaces. It reduces the gains that proper system integration architecture should provide. If external system "D" wanted to also obtain the payment information from external system "A", a whole new interface would need to be created.

The alliance instead develops a single, common structure for "Payment." We place data translation logic (for the peculiarities of each system) near the application adapters. The new solution follows in the diagram below.

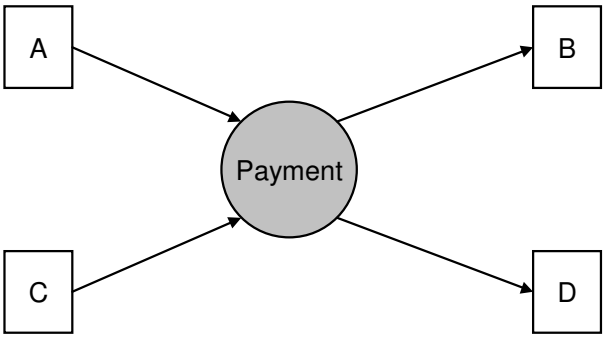


Figure 3C.12.8-8 Reuse-Driven Approach - *Our approach attempts to identify common business objects during design to reduce implementation and maintenance costs***.**

These common structures rely on our enterprise-wide data dictionary, described in Section 3C.12.6 Database Design and Development. We represent these common structures using XML. With XML, which is tag-based, the adapters easily use whatever elements of the message they require. They conveniently ignore other elements of the message. This contrasts with a fixed-length message, in which the adapter must parse through data elements it does not need. In this manner, the common message structures remain stable, while allowing each system the flexibility to only use what it needs.

3C.12.8.4.1.1.5 Apply Accepted Standards

There are several global standards when it comes to system integration, such as XML and Web Services. The alliance believes in adhering to stable standards when practical. Applying widely accepted standards "future-proofs" our integration with external agencies. It is more likely that these agencies will eventually migrate to these standards rather than to a proprietary data format.

Specifically, XML provides a great deal of flexibility when integrating multiple systems. Common message structures can be represented using XML and each system can use only the portion of the object they require. This means that common business object encoded in XML can grow and change without affecting the interfaces to the systems already using the object. This eliminates the need to coordinate across all external entities when we add a data item to a message to meet one entity's needs. Since we access data by markup tag and not position, the systems for those external entities not requiring the new data item simply ignore the new data item.

For example, the Credit Bureau's requirements for locate requests might change to include place of birth. We can add this field to the common locate request file without requiring CPLS and the other agencies to change their interface logic. Instead, they can

ignore the additional item indefinitely. Alternatively, they can incorporate this extra piece of information into their interface when it suits them.

As mentioned elsewhere in this document, we use XML for our internal messaging, but we do not force it upon other agencies. While our proposed eAI bus understands XML, it also translates for those external systems that do not speak the XML language. This use of eAI provides the CSE application the ability to restructure outbound messages to arrange, format, and transmit data in the same way as was provided to the external system before. Similarly, it provides the CSE application the ability to restructure outbound messages to match the way an SDU transfer system receives messages in other states. Likewise, the eAI bus transforms data received from the SDU vendor or other external system into the format desired by the CSE application.

For example, the ability to transform data into the format desired by the CSE application is vital to the maintainability of the IV-A interface. By using the data transformation capabilities of the eAI bus, the CSE application does not require separate, unique application code to process data received from each of the State's IV-A systems. Instead, the eAI bus insulates the application from these differences. It transforms IV-A case referral data into a single format understood by the CSE application logic that uses this IV-A information.

Lastly, an additional benefit to use of widely accepted standards is internal understanding and knowledge management. By applying a common data exchange format (like XML), future maintenance team members will likely know the standards. They will have less to learn to perform their tasks.

3C.12.8.4.1.1.6 Use Well-Formed Metadata To Define Interfaces

The alliance uses XML metadata interchange and the Unified Modeling Language (UML) to develop and validate the data exchange dictionary and rules. Rational Rose, IBM WebSphere Studio Application Developer, and IBM DB2 interact as a comprehensive environment to perform the data modeling, metadata creation, and XML Document Type Declaration (DTD) definition, management, and storage.

These structures are used to parse and validate data exchange transmissions with supporting application services to perform range, value, and case specific edits.

3C.12.8.4.1.2 Principles for SDU Integration

We recognize that the SDU is a critical piece of the overall CCSAS solution. We add the following principles unique to SDU integration:

1. Determine practical data exchange frequency
2. Implement audit and controls
3. Plan for recovery requirements
4. Anticipate interface definition changes
5. Allocate SDU performance requirements early

The following paragraphs discuss these principles.

3C.12.8.4.1.2.1 Determine Practical Data Exchange Frequency

The alliance's preferred approach is to design interfaces that support the exchange of information on demand and allow for the continual streaming of data. However, the alliance realizes that many external agencies may not be able to support the exchange of data on demand. In some situations, on demand data exchange may not provide any additional benefit to the program relative to the status quo. The alliance, the State, and external system staff evaluate the feasibility and need for on demand data exchange for specific interfaces.

The CSE application supports the receipt and processing of interface files from the SDU on demand. Likewise, the CSE application supports sending interface files to the SDU throughout the day. In doing so, the CSE application and SDU are better positioned to meet the federal requirement that collections must be disbursed within two business days of receipt. However, the CSE application also supports the scheduled exchange of data with the SDU, in the event the SDU vendor uses an existing commercial capability that cannot exchange data on demand.

3C.12.8.4.1.2.2 Implement Audits and Controls

To support data integrity, the alliance builds audits and controls into the SDU interfaces and external system interfaces. We use industry standard safeguards to confirm the accuracy of data received and processed through the SDU interface. This includes check summing, check digits, and transaction locking and integrity. We apply similar measures to other external system interfaces, where appropriate. SDU and external system interface reconciliation-processing checks and validates

- The completeness of transactions
- The usage of correct data element fields and names
- The usage of correct range and value for result fields
- The information in one transaction with that of a previous transaction

We suspend failed transactions or transaction batches, for correction or resubmission. The alliance anticipates that errors and error level thresholds are defined during the requirements and design process. These become part of the interface and transformation rules. This allows a varying level of automation and manual intervention in the data exchange process.

To supplement these controls, the CSE application also accepts information from the SDU to perform financial reconciliation. The SDU forwards to the CSE application information such as identified collections, unidentified collections, bank deposits, disbursements, checks returned, and other financial activities. The CSE application then crosschecks this information against other information in the CSE database. For example, for a given timeframe, the CSE application compares SDU deposits with the

total amount of identified and unidentified collections sent to the CSE application. In this manner, the CSE application financially audits message exchanges.

3C.12.8.4.1.2.3 Plan For Recovery Requirements

In the event of CSE application system failure, the SDU interface can recover using the eAI bus's journaling functions. The eAI bus provides a journal record, which can be stored to assist with disaster recovery. Tracking of message-level operations allows recovery from hardware failure without risk of data duplication. Independently of disaster recovery, the journal records also provide an audit trail.

3C.12.8.4.1.2.4 Anticipate Interface Definition Changes

Communication with the SDU vendor and external systems staff does not cease after designing, building, and operating. Child support is a dynamic program: Federal and State requirements change. Should such changes impact the definition of an interface between the SDU and CSE applications, the alliance works with the State and the SDU vendor through the alliance scope management change control process.

3C.12.8.4.1.2.5 Allocate SDU Performance Requirements Early

The alliance works with the State to define and allocate the performance requirements between the CSE application and the SDU. This includes the amount of time that:

- The SDU has to forward collection data to the CSE application upon receiving funds
- The CSE application has to process received collection data and return disbursement instructions to the SDU
- The SDU has to disburse funds upon receiving instructions from the CSE system

This allocation helps the CCSAS CSE application and the SDU together meet the federal two-day disbursement requirement.

We capture the requirements and their allocation for the SDU/CSE application interface in the Interface Design Document (IDD).[57] The allocation of the performance requirements early in system development allows both the alliance and SDU vendor to architect and size their hardware and software appropriately.

SDU/CSE performance requirements impact not only the application components that facilitate the transfer of data. They likely impact critical system functions that process or provide the data transmitted via the interface. For example, a performance requirement on the timeframe for processing collection data and returning disbursement instructions to the SDU may impact the design of the CSE application's payment distribution processing. This processing must handle collection data before disbursement

[57] See Section 3C.12.2 System Architecture Design for a discussion of the IDD.

instructions for the SDU can be generated. Many components of the CSE and SDU applications may be similarly and indirectly impacted by the allocation of SDU/CSE application performance requirements. Therefore, we define and communicate these requirements early. This reduces development rework.

The SDU vendor is responsible for developing its system to meet the interface and performance requirements set forth in the IDD. As such, the alliance conducts joint requirements review sessions during CSE/SDU integration planning. We review and clarify the interface and performance requirements allocated to the SDU application. In this way, the alliance and SDU vendor can identify and account for requirements that impact the implementation schedule—before attempting to coordinate SDU/CSE application integration activities. (If constraints prohibit the CSE-SDU integration from occurring as specified in the IDD, then the alliance will work with the State and the SDU vendor to develop a solution to the mutual agreement of involved parties.)

3C.12.8.4.2 Proven Work Products

The alliance brings existing work products based on experience with eAI implementations. These experiences and work products were formed across a wide range of eAI vendor products and client circumstances. The primary asset the alliance uses is the Architecture Delivery Toolkit, described below.

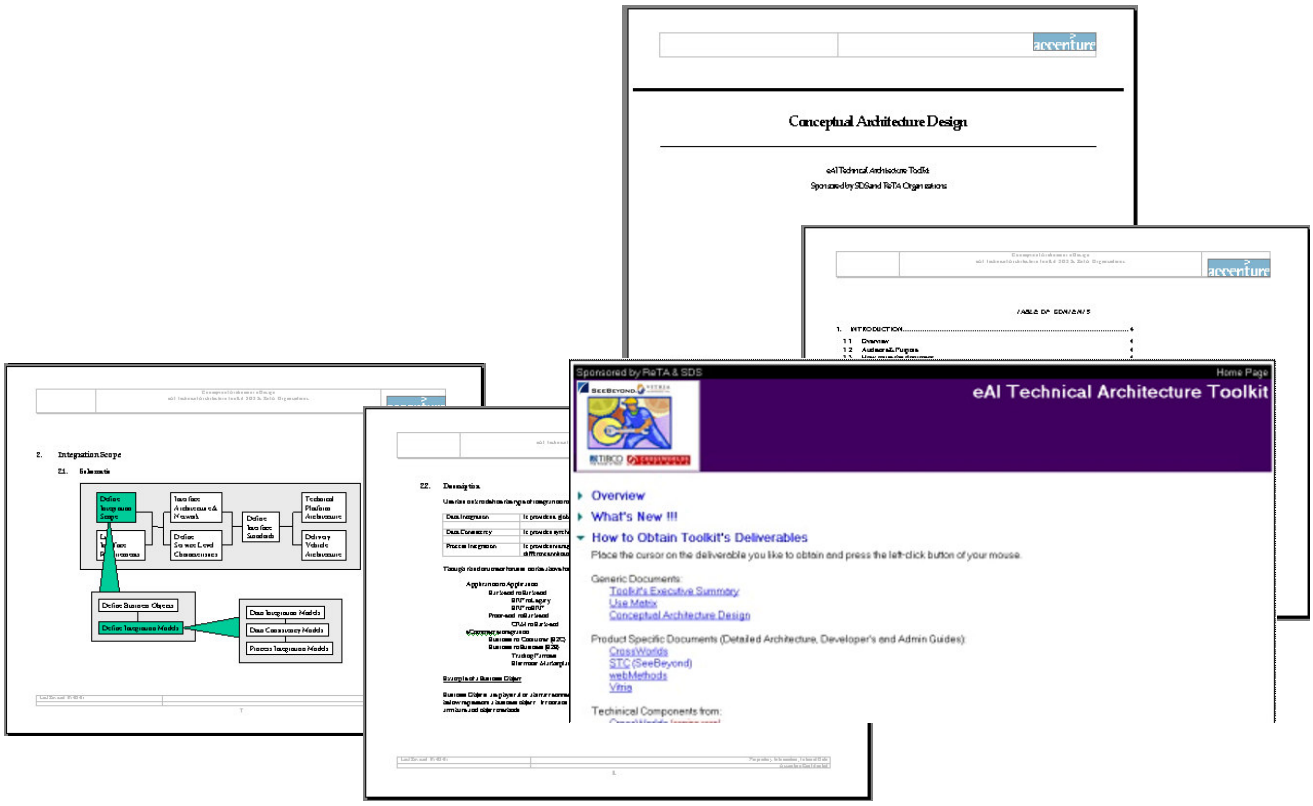


Figure 3C.12.8-9 Architecture Delivery *Toolkit -* *Our toolkit includes deliverables, job aids, and reference materials to effectively build the eAI infrastructure.*

3C.12.8.4.2.1 Architecture Delivery Toolkit

Accenture brings to the alliance its eAI Architecture Delivery Toolkit. This toolkit includes assets for designing and implementing the eAI infrastructure:

- **Use Matrix** – This vendor matrix represents substantial research across projects. We used this matrix to select SeeBeyond for CCSAS CSE needs
- **Conceptual Architecture Design** – This is the high-level architectural design for an eAI solution. The toolkit includes examples and templates that are portable across eAI messaging software
- **Detailed Architecture Design** – Also known as the eAI Framework, this design outlines the technical services and components necessary to support enterprise application integration. It serves as a blueprint for common services delivered by an architecture team
- **Developer's Handbook** – This includes SeeBeyond coding/naming standards, proven practices, developer processes, review procedures and a glossary of terms. It is analogous to the proven practices from the GRNDS Java architecture.[58]

[58] GRNDS is described in Section 3C.12.4 Software Architecture Design.

- **eAI Administration Guide** – This guide is a detailed list of required day-to-day activities
- **Technical Component Library** – This library contains reusable code and documentation from other SeeBeyond projects. The code has been tested and built for reuse

Our contract provides a no-cost, irrevocable license to the CCSAS project for this toolkit.

3C.12.8.4.3 Proven Process

A key feature of the alliance's solution is that system integration follows the same overall approach as general system building. Having a completely different approach would mean developing software components and integration components along different paths, potentially with coordinated review points along the way. Ironically, this would make it more difficult for integration components to **integrate** with the very software components that send and receive the system messages.

And there is no need for a different approach: ultimately, system integration is recognized in software, as record formats, as validation scripts, and as routing rules. Granted, there are differences in the architecture approaches, design patterns, tools, deliverables, scripting (programming) languages, and required team skills. But the *overall* methodology steps – from requirements analysis through design through testing – remain the same.[59] In this manner, for system integration, we:

- Arrive at a comprehensive, consistent, and complete set of requirements
- Design architectures and software that satisfy requirements
- Incorporate CCSAS architectural attributes and proven practices into the requirement baseline and into system integration designs
- Define and validate behavioral and performance requirements
- Perform knowledge management activities for smooth transition to CCSAS project staff

…just as we did for our online and batch components, coded in Java.

For but a few examples:

- When we identify CCSAS decision makers for the system architecture, we do this not only for hardware and component software, but also for system integration[60]
- We build technology prototypes to validate behavioral and performance requirements for system integration, as we do for other system requirements[61]

[59] This includes System Requirements Analysis (Section 3C.12.1), System Architecture Design (3C.12.2), Software Requirements Analysis (3C.12.3, only now "software " refers to logic for system integration), Software Architecture Design (3C.12.4), Software Detailed Design (3C.12.5, for the detailed design of eAI scripts), Software Coding, (3C.12.7, for the coding of said scripts), and Test Management (3C.12.9).
[60] See Section 3C.12.1 System Requirements Analysis
[61] See Section 3C.12.2 System Architecture Design

- When we talk about testability as a litmus test for detailed software requirements, we apply it as well to system integration[62]
- When we use the "V" model and use the software design to validate the requirements, we do this for system integration as well[63]
- With system integration detailed design, as with software development, we conduct kick-off meetings with SMEs to expect (rather than inspect) quality of designs[64]
- We follow a similar process for informal code reviews of system integration scripts, by inviting the same types of SMEs, who use a similar process to review for compliance with standards CCSAS architecture attributes[65]

[62] See Section 3C.12.3 Software Requirements Analysis
[63] See Section 3C.12.4 Software Architecture Design
[64] See Section 3C.12.5 Software Detailed Design
[65] See Section 3C.12.7 Software Coding

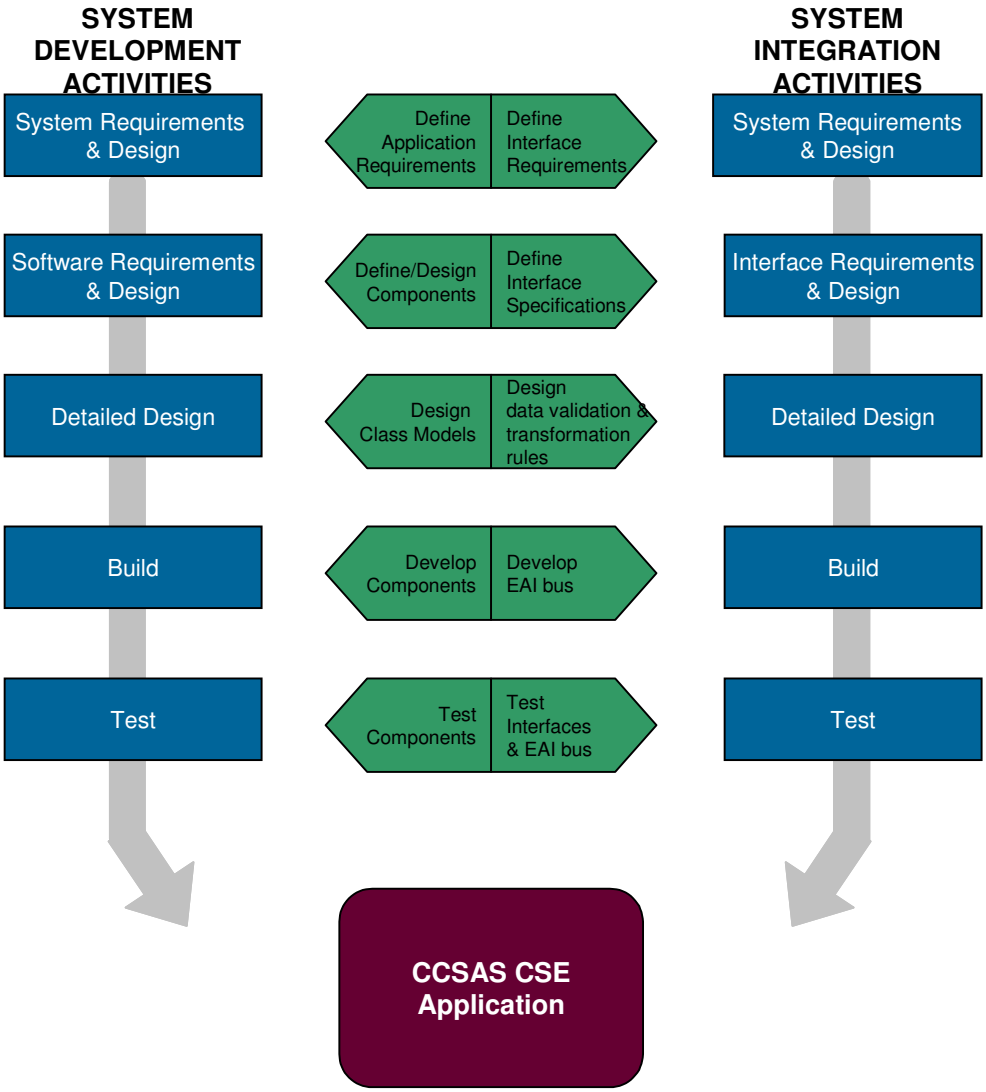


Figure 3C.12.8- 10 Relationship between System Development and System Integration Activities - *Activities in these two development processes mirror one another.*

In this manner, we need not invent yet another methodology for system integration. Moreover, **uniformity in overall methodology gives us natural, ongoing touch points between building system integration and software building.** In sum, shared methodology helps us design, build, and test system integration pieces that interface properly with the CSE application components – components that integration depends on.

This methodology uniformity is also influenced by the alliance's *architected* approach to eAI. This is a key part of our solution. Many other consulting firms think of integration as simply message formats and queues for passing around messages. These firms code much of the logic related to messaging into the application components that send and receive messages. That is, they make the application module responsible for

formatting, validating, transforming, and routing a message onto a passive integration bus. Under this typical approach, integration activities require a simplistic methodology, as they involve configuration more than they do any substantial coding.

In marked contrast, the alliance proposes a layered approach. This clearly separates integration code out from application code. Recall the layered architecture from Section 3C.12.2 System Architecture Design, as shown in this diagram:

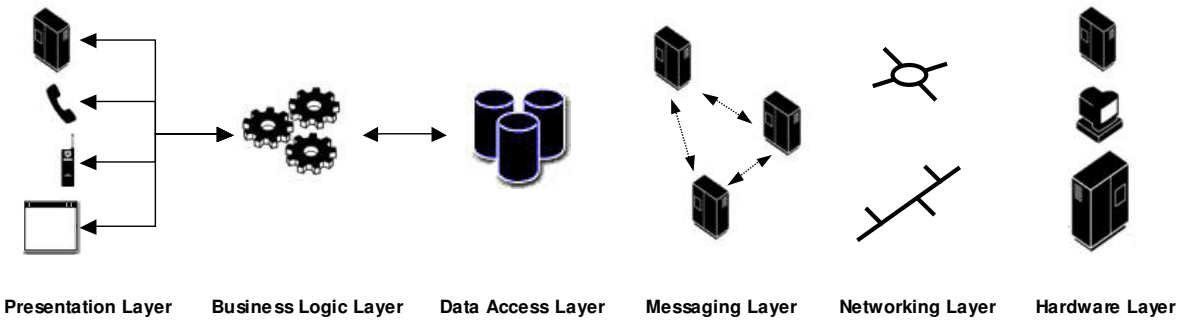


Figure 3C.12.8-11 Architecture Layers - *System Integration has its own architecture layer, to simplify the fundamentally complex space of large systems building.*

In this way, the alliance builds formatting, validation, transformation, and routing logic into their own eAI modules. We can then change integration rules independently of the business applications. This provides maintainability and extensibility, directly addressing many of the State-specified CCSAS architecture attributes.

Our more robust eAI architecture requires a more robust methodology, that includes tasks for eAI detailed design and coding. This is another key reason for sharing the same overall methodology between software and integration development. We simply reuse many of the principles and methodology steps that we have already created for object-oriented code.

3C.12.8.4.4 Proven Tools

We separate the overall integration architecture into the following logical layers:

- **Messaging** – provides network-based locations (i.e., queues) where applications can place data for the purpose of communicating with other applications
- **Application integration** – provides logic on how to communicate with applications, and how data from these applications should be formatted, routed, and transformed
- **Enterprise workflow management** – provides logic to support large, long-lasting transactions that require message persistence and manual intervention

- **External partner management** – provides logic to support communication with other enterprises using standard communications protocols and data formats (e.g., EDI, XML, SOAP)

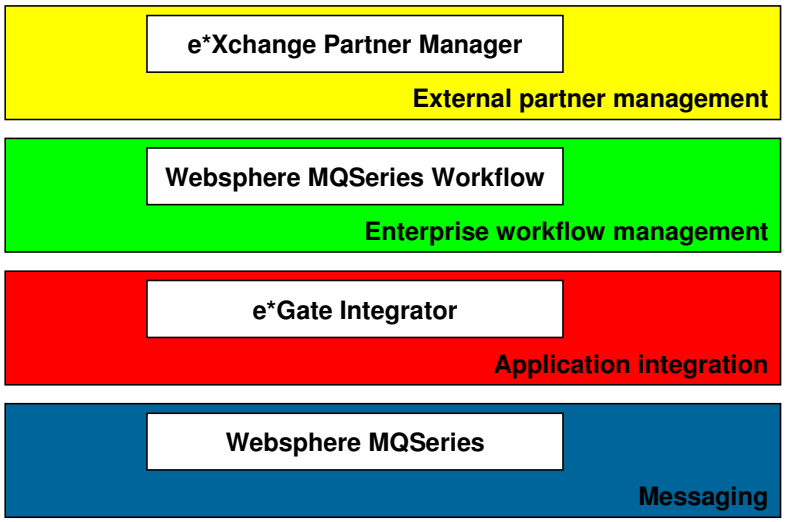


Figure 3C.12.8-12 General Product Architecture - *Our proven tools and products include various top vendors and a layered approach.*

The following paragraphs describe the products we have selected for each logical layer.

3C.12.8.4.4.1 IBM Websphere MQSeries

MQSeries is market-leading, integrated middleware for connecting applications, in and across organization boundaries. Its base messaging servers and clients provide once, and once only, message and queuing capabilities on over 35 platforms. Its features include:

- Heterogeneous, any-to-any connectivity from desktop to mainframe
- A comprehensive family of APIs designed to simplify coding for messaging tasks
- Asynchronous communication
- Publish-and-subscribe communication
- One-time delivery

3C.12.8.4.4.2 SeeBeyond e*Gate Integrator

e*Gate Integrator handles the logic related to intelligent routing, validation, formatting, and transformation of messages. This market-leading product connects and manages information across a number of enterprise applications. Its features include:

- Open, distributed architecture that provides an extensible framework for integrating applications, systems, and partners by leveraging leading standards (such as XML)

- A graphical data transformation tool that generates integration logic based on standard programming languages, including Java
- Component-based, off-the-shelf packaged connectivity solutions for major business applications (including Siebel), legacy systems, Web, and middleware environments
- Management and monitoring of the distributed environment from a central console

3C.12.8.4.4.3 IBM Websphere MQ Workflow

MQSeries Workflow is a market-leading business process management system. It supports the rapid development and management of the business processes that integrate the IT and organizational infrastructure of an enterprise. Its features include:

- Model-driven e-business process automation and tracking
- Transactional and universal integration based on MQSeries and XML: messages do not get lost nor executed twice
- Performance that scales with growing workload needs and server capacity
- Process tuning based on audited experience and changing business needs

3C.12.8.4.4.4 SeeBeyond e*Xchange Partner Manager (PM)

Built on the e*Gate Integrator platform, PM allows for easy maintenance of trading partner profiles. We use it to build a trading partner network of external agencies and companies. Its features include:

- Intuitive web-based interface for managing trading partner profiles in the partner repository. This includes information on communication protocols, security keys, message locations (e.g., FTP addresses), and agency contacts
- Rules-based protocol engine, for flexibility in fine-tuning individual trading partner parameters
- Support for leading eBusiness protocols including RosettaNet 1.1/2.0, EDI X12, EDIFACT, CIDX, NCPDP and BizTalk
- Web-based transaction auditing capability, including transaction reconciliation, message searching and error viewing
- Comprehensive public key infrastructure (PKI), for end-to-end transaction security

3C.12.8.5 Managing External Entities in Common with the SDU

This section describes our approach for managing interfaces with external entities in common with the SDU.

When the SDU and CSE application need the same data from an external agency, the CSE application shall be deemed the “system of record”. We believe that the SDU data capture requirements will be a subset of the CSE application data capture requirements. The CSE application can therefore own this data.

Having the CSE application forward common data to the SDU has the following advantages:

- Eliminates the need for external entities to interface with both the SDU and CSE applications to provide the same data. This simplifies the coordination of changes since the external entity interfaces with only one application instead of two.
- Reduces disruptions to the CSE application by reducing the CSE application’s reliance on the SDU vendor’s application for external entity information.
- Reduces changes to the SDU vendor’s system, should the CSE application require additional information from an external entity.

This approach does not eliminate or discourage coordination with the SDU vendor. Rather, the CSE application’s role as conduit for common external data fosters coordination. Because the CSE application retrieves data from external entities and forwards it to the SDU, we need to consider the SDU’s informational requirements when changing CSE external interfaces. Prior to modifying a CSE external interface, we assess the impact of the other CSE functions that use the data provided by this interface. This includes assessing those CSE functions responsible for forwarding external entity data to the SDU. This allows us to resolve issues and agree upon a design with the SDU vendor, prior to modifying an external interface.

When the SDU builds interfaces with external entities that do not have interfaces with the CSE application (e.g., interfaces dealing with payment receipting), we expect the SDU vendor to include the alliance during the interface design process so that CSE data needs are addressed.

3C.12.8.6 Integrating CSE System Hardware and Software

Our approach to integration subjects the technical infrastructure, technical architecture, and CSE application functions – including interfaces – to multiple levels of testing prior to deployment.

The following paragraphs discuss the approach, methods, and tools we employ to accomplish this.

3C.12.8.6.1 Testing Technical Infrastructure and Architecture Integration

Technical infrastructure and architecture testing focuses on testing the physical network, computing devices, execution architecture, and operations architecture that support the CSE application and its interfaces with external entities.[66] The components of the technical infrastructure and architecture undergo multiple tests. Following the "V" model,[67] each subsequent test validates a higher level of integration. The alliance performs the following tests on technical infrastructure and architecture components, as appropriate:

- Unit Test
- Unit Integration Test/Technology Architecture Test
- System Test

Unit testing of the physical network and computing devices is limited to the computing devices and any associated network media and transport services. These tests verify the computing devices and any associated network media and transport services are installed, configured, operational, and accessible. Upon completing unit testing, the physical network and computing devices undergo unit integration testing. This verifies that physical network and computing devices interact properly with each other.

Upon completing unit and unit integration testing for the physical network and computing devices, we test the services of the execution and operations architectures. Unit testing for the execution and operations architectures validates that custom developed components and extensions to reused components meet the requirements documented during technical architecture design.

Following unit testing, we construct a reference application and conduct a technology architecture test/unit integration test. The reference application and the technology architecture test validate that custom, reused, and packaged components of the architectures provide the required services, integrating as specified in the architecture design.

System testing of the technical infrastructure and architecture components is performed in conjunction with system testing of the CSE application prior to implementation. We conduct system test in the testing environment. In addition to testing the integration of the application, architecture, physical network, and computing devices, system test also includes performance testing. Using the production simulation environment, the alliance validates the infrastructure, architecture, and application performance requirements.

The alliance automates testing where appropriate. See Section 3C.12.9 Test Management.

3C.12.8.6.2 Testing Application Integration with External Entities

[66] Section 3C.12.2 System Architecture Design defines execution and operations architecture.
[67] Section 3C.12.9 Test Management defines the "V" model.

CSE interfaces (including data translation rules) with the SDU and external entities undergo the same levels of testing applied to other CSE application functions prior to deployment. As noted in Section 3C.12.9 Test Management, these levels include:

- Unit testing
- Unit integration testing
- System testing

In testing the SDU and external interfaces, the alliance relies on the same tools, processes, and procedures described in Section 3C.12.9 Test Management. Our approach positions us to identify, contain, and repair defects early in the development process. This avoids rework for the alliance, the SDU vendor, and interfacing agencies.

As described earlier in this section, our approach to designing, building, and testing CSE interfaces includes coordination with the external interfacing agencies and SDU vendor. Prior to conducting system testing, the alliance coordinates testing with the SDU vendor and external agencies, with the State's assistance. The alliance provides external agencies and the SDU vendor with transaction files for testing. We similarly expect the SDU vendor and external agencies to provide files for testing in the opposite direction.

As mentioned, the alliance automates testing as appropriate. See Section 3C.12.9 Test Management.

3C.12.8.7 Conclusion

The alliance's integration solution is differentiated by our architected approach for the messaging layer. Our robust integration architecture encapsulates the logic for message routing, formatting, translation, and validation in an eAI bus. This allows us to maintain messaging logic separately from business logic, giving us flexibility when messaging requirements change.

As well, we use industry standards, like XML, for maintainability and for better positioning the CCSAS project for future technologies. We define common message structures, using our enterprise-wide data dictionary, for reuse. Lastly, we use proven, market-leading tools, from SeeBeyond and IBM's MQ family.

The alliance brings to the CCSAS project significant experience with eAI technologies and approaches. We apply innovative principles, like using eAI components, which other client engagements have proven in production. We also reuse design and code assets from these engagements, thereby giving us a jump-start on the CSE integration architecture.

Our process for system integration is robust and straightforward: we reuse approaches and steps from our proposed system development methodology. This uniformity in overall methodology gives us natural, ongoing touch points between building system

integration and software building. The shared methodology helps us design, build, and test system integration pieces that interface properly with the CSE application components – components that integration depends on.

But integration with external entities is more than process, principles, and tools. It is also about communication. We understand that, and we understand in particular the importance of successful collaboration with the SDU vendor. We have clearly defined roles and responsibilities for integration of the CSE and SDU applications. We coordinate CSE and SDU schedules, we hold joint design meetings, we review each other's work, and we resolve problems together. We simplify the management of external interfaces in common with the SDU, by owning the common data and forwarding it to the SDU. This communication – along with the processes, principles, and tools – brings integration success to the CCSAS project.

3C.12.9 Test Management

We leverage our experience and robust testing approach to build a comprehensive and effective test plan. We focus testing efforts to maximize coverage and "ferret out" defects. We understand the new techniques required to test components. Lastly, we use metrics throughout to track progress, predict trends, and identify areas for improvement. For the alliance, testing is not an afterthought.

3C.12.9.1 Introduction

Some organizations wait for their official testing stage to kick off in order to begin testing. Simply, they wait until build is almost done to plan, they wait to create test conditions, they wait to create expected results. We believe testing begins much earlier.

We use proven methods and tools for planning and conducting tests and documenting and analyzing test results. This section includes:

- Overview of our testing approach
- Approach for test coverage
- Different layers of tests and our use of stage containment
- Management of test data, including identifying sources of test data and obtaining, preparing, and maintaining that data
- Approach for testing interfaces with the SDU and with external agencies
- Methods and tools for regression testing
- Methods and tools for performance testing
- Approach for independence in testing
- Integration of test outcomes into software build planning activities
- Integration between testing and our problem resolution process
- Approach for supporting State System Qualification Testing

We use a robust and repeatable testing process, automating where appropriate.

A robust test plan includes multiple layers of testing for stage containment. We test at these multiple layers, and we use automation and tool support where appropriate.

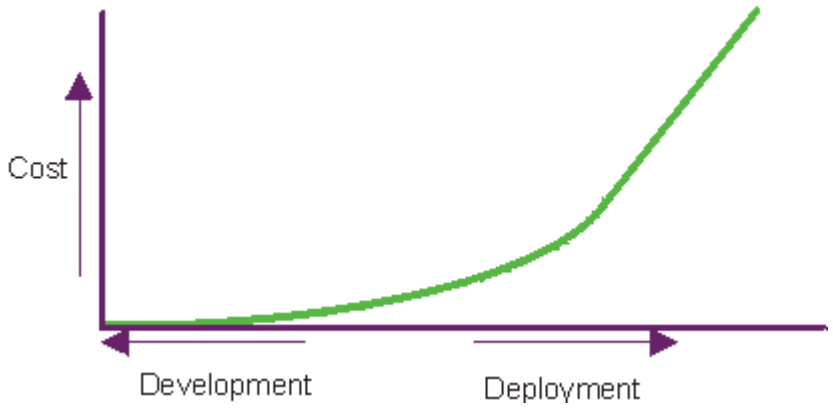


Figure 3C.12.9-1 Importance of Testing - *Software problems are 100 to 1000 times more costly to find and repair after deployment.*[68]

Studies have shown that defects discovered later in the software lifecycle cost more to fix. Defects discovered later have a greater impact on the schedule. Our process focuses on detecting defects early.

Test early, and test often.

3C.12.9.2 Proven Principles

This section describes methods and tools for planning tests, conducting tests, and analyzing results. It also discusses automated testing and debugging tools.

An effective testing approach requires:

1. Early, up-front planning
2. Creation of test packages based on business cases
3. Well documented and repeatable test cases
4. Tight control of the testing technical environment
5. Extensive communication
6. Integrated testing organization
7. Extensive use of testing tools where appropriate
8. Traceability of test plans to requirements
9. Stage containment

The following paragraphs discuss these key concepts.

[68] From the Rational Unified Process

3C.12.9.2.1 Early, Up-front Test Planning

Proven practices dictate that teams should perform test planning early on. This contributes to a more thorough test. Our planning for each testing stage produces:

- **Test Conditions.** These describe the comprehensive set of conditions for testing the application.
- **Test Scripts.** These define the steps, input data, and expected results for a number of test conditions. We define scripts for each level of testing – Unit, Unit Integration, and System Test.
- **Test Cycles.** These plan each cycle, within the context of the larger test.

We execute tests at a variety of levels during system development. Early test execution shortens the overall schedule by catching, isolating, and fixing bugs before they have ripple effects on other modules. The CCSAS CSE project will have six types of testing (these are discussed in Section 3C.12.9.4.2 Multiple Levels of Testing). These tests include the following:

- **Unit Test** – This test is planned and performed by the coder. Its purpose is to test an individual component.
- **Unit Integration Test** – This test is planned and performed by the coder or another member of the Application Development team. Its purpose is to test that intimately related components interact correctly.
- **System Test** – This test is planned and executed by the CCSAS CSE testing team. It tests that components within a capability group function to perform the business processes that the application supports. It also tests that the components across capability groups integrate properly to perform the business processes that the application supports.
- **System Verification Test** – This test is planned and executed by the CCSAS CSE testing team, with increased involvement and participation from State CCSAS Project representatives. Its purpose is to demonstrate the proper execution of the application prior to commencement of the State's formal testing efforts.
- **System Performance Test** – This test is planned and executed by the CCSAS CSE architecture team. Performance testing is conducted informally throughout system development activities. A formal System Performance Test concludes this effort.
- **System Qualification Test** – This test is planned and executed by State CCSAS Project representatives. Its purpose is to confirm that the system meets user acceptance and performs the business functions required.

3C.12.9.2.2 Test Packages Based on Business Cases

We base integrated test packages on realistic business cases (i.e., Use Cases defined in the system and software analysis and design stages). We involve key CCSAS subject matter experts early on. We continue to work with them throughout the testing process to create relevant and effective tests. System Test focuses on confirming that a job function can be performed with the application. This gives us confidence that the application supports the business functions being tested.

Reuse calls for regression testing, to check that module changes do not impact everyone using that module.

3C.12.9.2.3 Repeatable Test Cases

As changes affect the environment, data, and software, the application must be regression tested in a robust fashion. Fixing a bug in one module may have the unintended consequence of breaking another module. We document test cases to achieve:

- Thorough and consistent testing, for a high quality application
- Reduced time spent on testing activities because developers need not repeatedly determine what to test

We focus on test repeatability. A test case is a set of test data, scripts, and expected results that can be automated in a testing tool. These test cases are developed during test planning. We periodically update the test conditions, scripts, and cycle control sheets that comprise these test cases. This keeps tests up-to-date. This facilitates regression testing, which is particularly important in a component environment. For, reusing code also means reusing defects. But having repeatable tests encourages the team to test changes in a broad fashion. This means testing not only the changed module, but also other code that reuses that module.

3C.12.9.2.4 Tightly Controlled Technical Environment

We tightly control our technical environment. The alliance manages test environments including test data and deliverables using the configuration management process discussed in Section 3C.4 Configuration Management. During System Testing, System Qualification Testing, and various levels of Performance Testing, we test components in a “production-like” environment.

Stable test environments help us quickly isolate bugs.

A stable environment reduces the number of defects introduced by a changing, ad hoc environment configuration. Moreover, when defects are found in a stable environment, developers can conclude that the defects are due to application problems rather than environment concerns. This avoids wasting time by properly focusing the investigation.

3C.12.9.2.5 Extensive Communication

We communicate extensively to better manage the test effort and to confirm that it meets its goals. Our open communication includes regular status meetings and visible progress charts and metrics. We provide written and oral communication for both the testing and the fix-it teams.

Prominently displayed progress charts motivate and focus the testing effort.

Communication helps focus our priorities on fixing important bugs early. Only when the fix-it team understands the big picture can it appropriately reassign priorities. This might include putting more experienced resources on a particularly difficult or "buggy" component.

3C.12.9.2.6 Integrated Organization

An integrated testing organization is a proven principle of success. There are a number of ways the alliance promotes integration, including:

- Comprehensive testing from a functional and business perspective is accomplished by having functional users and Subject Matter Experts (SMEs) involved in both test planning and verification.
- Representatives from the CCSAS project staff should be allocated to teams to promote knowledge transfer to State staff. This also provides opportunities for alliance testing staff to gain exposure to State best practices.
- We leverage skilled application developers to increase testing team productivity. Knowledge about the application and the tools used for development and testing can be used to more effectively and efficiently manage problem/defect resolution.
- Testing responsibilities for team members that came from application development should be limited. Specifically, we limit their roles in planning and verification.

With these guiding principles in mind, we organize the testing team into cells. For the System Test, the testing cells are capability (functional) based. For the test planning stages these teams include Subject Matter Experts who were involved in analyzing software requirements as well as Senior Business Analysts who are familiar with the testing tools and processes. These integrated teams of alliance and CCSAS project staff plan the testing effort.

Application Development team resources begin to transition into testing roles to prepare test data and help setup test tools. Once test execution begins, additional resources from Application Development join test cells to resolve defects. This allows the testing cells to control their own success, instead of throwing defects 'over the wall' to a separate defect resolution team.

3C.12.9.2.7 Testing Tools

Where available and appropriate, we use COTS (Commercial Off-The-Shelf) tools for documenting test cases, tracking issues, recording and playing back scripts, generating and manipulating test data, comparing actual and expected results, and managing test configurations. Automation with tools simplifies the testing process, especially for large, complex applications like CCSAS CSE. We have found the following types of tools to be particularly important:

- **Problem management.** Tools for tracking verification and validation problems encountered during test execution.
- **Functional test execution.** Tools for recording test scripts, executing playback, and comparing expected with actual results. Scripting should support functional testing as well as stress and performance testing. Automation in this area really simplifies test regression, which can be executed "at the push of a button".
- **Performance test execution.** Tools for creating volumes of test data; loading the system; measuring throughput, response time, and other performance characteristics; and identifying performance bottlenecks.
- **Test planning.** Tools for documenting test conditions, test cycles, test scripts, test data, expected results, and test plans.

To address TM 12.9.8, we use Mercury Interactive Test Director and JUnit to support test planning and automated, repeatable testing. These COTS packages support testing at different levels. Both manage test cases and scripts. We use other COTS tools for problem management.

3C.12.9.2.8 Traceability

We use DOORS for requirements management. DOORS integrates with Test Director, introduced above. This integration allows one to create electronic links from a particular detailed requirement in DOORS to a particular test condition in Test Director. This traceability allows business analysts to see that detailed requirements have been tested.

As described in 3C.12.1 System Requirements Analysis, summary requirements describe requirements in a manner too broad to be tested. Therefore, our testing approach focuses on traceability to and the testing of detailed requirements.

Test Director also supports traceability by linking defects to:

- The test scripts that first discovered them
- The specific versions of the specific modules that contain the resolution to the defect

We use any such out-of-the box capabilities of our tools for this traceability.

3C.12.9.2.9 Stage Containment and the "V" Model

CCSAS CSE's complexity requires a structured test process with stage containment. The purpose of stage containment is two-fold:

- Stage containment defines the scope of a test to a finite set of requirements
- It identifies problems in the developing product before they pass to the next stage

Stage containment promotes quality in the testing process and the end product, the CCSAS CSE system. The goal of stage containment is reducing the number of problems being passed to the next stage.

Stage containment catches errors before they bleed into the next stage. This builds quality into each stage.

The industry's "V" Model prescribes stage containment throughout the software lifecycle. This model breaks testing into stages that are tightly focused on different aspects of the application. It also incorporates numerous reviews for thoroughness and quality.

The "V" Model provides consistency and compliance with IEEE standards. "V" model benefits include improved quality and reliability, reduced rework, reduced time spent on correcting problems, and more efficient testing. It focuses the objectives of each test based on where we are in the development lifecycle.

The model relies on three techniques:

- **Verification** checks that a work output (e.g., code module) is correctly derived from the inputs of the corresponding stage. It also confirms that the output is internally consistent, conforming to quality standards. Verification techniques include:
- **Desk checks** where the developer inspects the work output against the verification exit criteria.
- **Walkthroughs** where the developer and a supervisor or peer inspect the work product against the verification exit criteria.
- **Validation** checks that the work product satisfies the requirements specified in an earlier work product. It also checks that the Business Case continues to be met – that the work product is within scope, contributes to the intended benefits, and does not have undesirable side effects.
- **Testing** checks that a specification is properly implemented. Ideally, testing only uncovers problems made in translating the specifications into the product, rather than problems in the specifications themselves. As can be seen in Figure 3C.12.9-2, the V model starts with high-level project requirements and progresses to detailed specifications and coding. When starting back up the right hand side of the V model, testing begins at the granular level, the Unit, and then broadens back out to test the software and finally system requirements of the application.

Verification confirms the program is built according to its specifications.

Validation confirms the specifications indeed meet requirements.

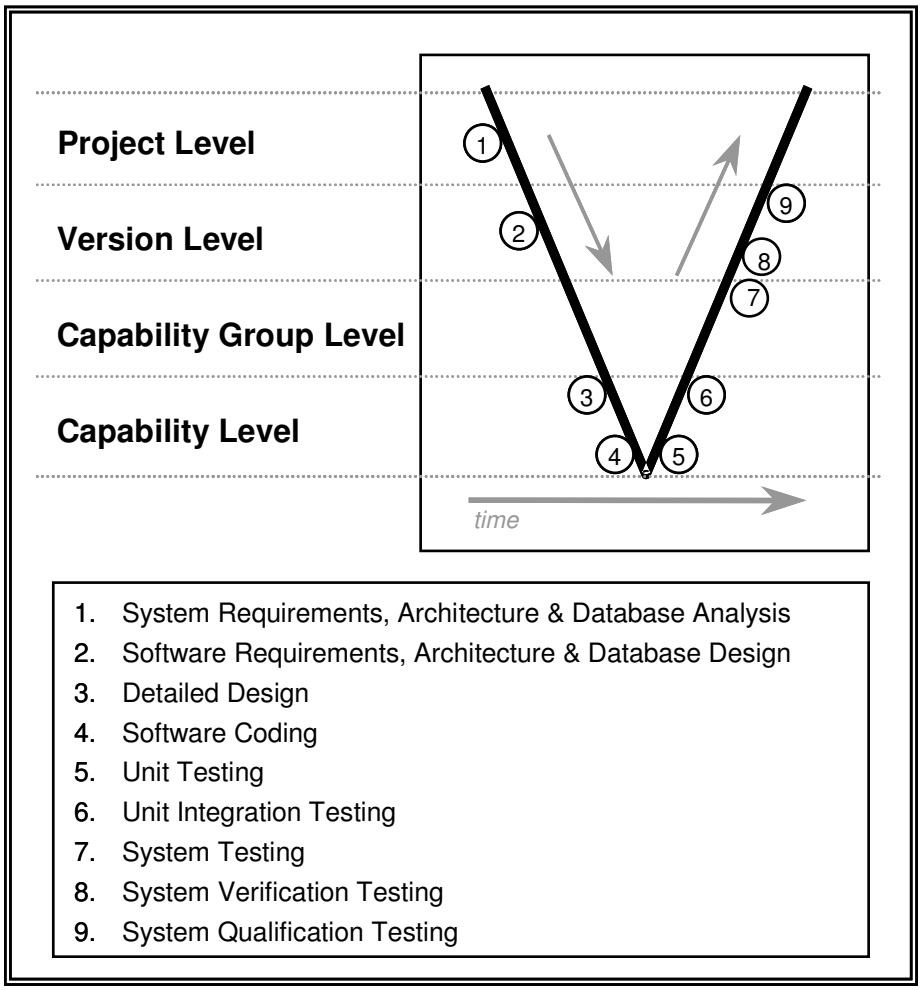


Figure 3C.12.9-2 V-Model - *The V-Model provides a thorough stage containment structure.*

The "V" model structures the development to deliver quality at each stage in the process.

The V Model offers guidance on test scope. Unlike the left side of the model where each step builds on the previous (e.g., the detailed design specification provides more details about how to implement the software requirements), the right side does not build on the prior test step. Instead, proper test plans are based on the specifications and requirements that correspond on the left side of the V model.

For example, a proper system test plan is not based on summarizing the preceding Unit test and Unit Integration test. Rather, it is instead based on the left side of the model, the software requirements. This fundamental aspect of the "V' Model results in a test at each level that focuses on a set of requirements, without being redundant with the preceding test.

3C.12.9.3 Thorough Test Coverage

The Rational Unified Process states, "While the complexity of software makes complete testing an impossible goal, a well-conceived methodology and use of state-of-the-art tools can greatly improve the productivity and effectiveness of the software testing."

We use thorough test coverage for delivered functionality and code paths. For proper coverage, we:

- Establish guidance and standards for the test effort
- Define entrance and exit criteria for each test
- Establish an internal review and approval process
- Trace test plans to requirements
- Use metrics to evaluate progress and success

Test coverage helps us catch errors at the appropriate stage – supporting stage containment.

The following paragraphs describe each in more detail.

3C.12.9.3.1 Guidance and Standards

We establish guidance and standards through the Master Test Plan (TM 032) and Software Test Plan (TM 033). These guide the Test Team in its activities, helping it to conduct tests that are consistent and comprehensive.

3C.12.9.3.2 Entrance and Exit Criteria

The Master Test Plan deliverable (TM 032) defines the entrance and exit criteria for creating and executing each test type. Example criteria include:

- The proper development process was followed and each step of the process was correctly executed
- The test plan was correctly derived from inputs
- Coverage was verified by assessing the test plan relative to branches in the program code or business process
- Comprehensiveness was verified with traceability of the test plan to the requirements
- Test data and database activities for establishing test data were verified
- Tests conformed to project testing standards set forth in the Master Test Plan regarding content and format

We use clearly defined criteria to determine test completion.

3C.12.9.3.3 Internal Review and Approval Process

Internal review and approval steps are integral parts of the "V" model. Reviews help confirm the thoroughness of test plans and their conformance with standards. Reviews also clear up miscommunications about the process in a timely manner.

Reviews are conducted first by developers and their peers. They verify and validate the work product against test exit criteria. Subsequent reviews include a supervisor or

subject matter expert. This person should have a working knowledge of the inputs (e.g., functional requirements, designs, performance requirements) that the test plan validates.

Internal reviews are conducted:

1. **At the conclusion of test planning.** These reviews verify and validate the test conditions, scripts, and cycle sheets that comprise the test plan against the exit criteria set forth in the Master Test Plan. Should the test plan not satisfy exit criteria, the reviewer clarifies deficiencies so that corrections can be made. Once corrected, the test plan is resubmitted for review. This cycle continues until the exit criteria are met.
2. **Prior to test execution.** These reviews evaluate the entrance criteria that must be satisfied prior to executing a particular test. The reviews verify and confirm the inputs to test execution are in place. Inputs include the software/hardware to be tested, test conditions, test scripts, and test cycle sheets. The reviews also confirm that the test team has obtained any approvals required prior to testing.
3. **After test execution.** These reviews focus on exit criteria related to test outcomes and test objectives. The reviewer examines the documented test results as well as the test plan to verify the successful execution of the test. The reviewer also validates the test objectives were accomplished prior to progressing to the next test stage. Should the review uncover an issue with the test execution, corrections are made and the test re-executed.
4.

3C.12.9.3.4 Test Plan Traceable to Requirements

We derive test cases for System Testing from:

- Use cases, representing functional requirements
- Technical requirements for architectural attributes, behavioral requirements (e.g., performance and usability), and other categories as defined in the Master Test Plan

Furthermore, we establish and maintain traceability between the test cases and requirements to help us create comprehensive test plans. Telelogic DOORS, our requirements tool, provides a module that gives traceability into test scripts written in Mercury Interactive Test Director. This traceability confirms that each detailed requirement has an associated test. Passing the tests means that the application has met its detailed requirements.

3C.12.9.3.5 Metrics Gauge Success

Our approach includes metrics, which measure coverage of the testing effort. Coverage can be expressed in terms of:

- Percentage of requirements or test cases that have been tested. This is useful in later test stages

- Percentage of code executed, which is useful in unit test stages

Metrics also measure quality in terms of reliability, stability, or performance. We measure quality by looking at the number and types of defects found during the testing process.

We use the defect management tools built into Mercury Interactive Test Director. This market-leading testing suite provides powerful defect management functionality. It includes built-in reports for a number of different testing metrics. We use the COTS capabilities of our tool to gauge progress and success. This knowledge guides the testing and defect correction processes.

Metrics give us a handle on the testing situation for appropriate action.

Metrics provide several benefits:

- **Support management based on facts.** Metrics provide objective data upon which decisions can be made, actions can be taken, and goals can be achieved.
- **Encourage focusing on the problem, not the symptom.** Metrics facilitate a better understanding of problems. For example, if application product test is behind schedule, people might automatically assume that the original estimates were bad. Useful metrics, however, can point to the truth. There may be a number of reasons for being behind schedule: vague requirements, code that was not unit tested, or an unstable environment.
- **Enable predictability.** We collect appropriate metrics accurately and consistently. Then we can extrapolate from the metrics, to predict the quality and productivity of the remaining work, the next development stage, or the next release.
- **Support continuous improvement.** Throughout the testing process, we repeatedly review the metrics, gain an understanding of problems, and improve the process or the estimating guidelines. Our incremental approach allows us to learn from metrics from an early increment, to improve development and testing on subsequent increments.

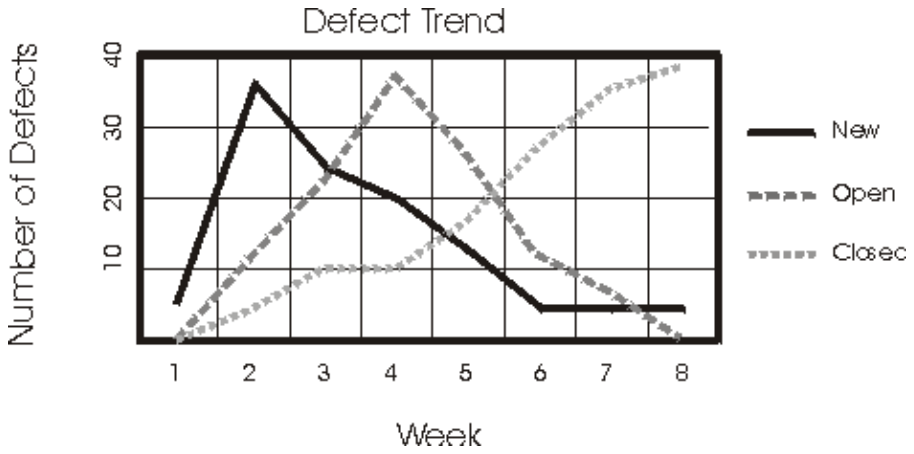


Figure 3C.12.9-3 Testing Reports - *Defect reports help measure quality and progress by extrapolating from trends.*

Throughout the testing process, we use the tool to periodically distribute reports. Depending on the tool's particular capabilities, reports include:

- **Defect counts,** like the number of defects with a particular state of "open" or "closed," number of defects with a particular priority, number of defects with a particular severity, or number of defects from a particular component.
- **Defect trends,** like the number of defects open or closed as a function of time. In particular, this allows us to measure the sweet spot in testing when the number of defects closed per day is greater than the number of new defects opened. Finding this "over the hump" point helps us accurately predict when the testing stage will end.
- **Test progress,** like the results of testing over a number of test cycles.

3C.12.9.4 Structured Test Approach

Our structured test approach includes multiple levels of testing, conducted according to well-defined test plans. The following paragraphs identify the plans and the types of tests that we perform.

3C.12.9.4.1 Test Planning

Test planning guides test activities. The Master Test Plan (TM 032) provides an overview of the testing effort while the Software Test Plan (TM 033) provides more test implementation details. The following paragraphs describe these plans.

3C.12.9.4.1.1 Master Test Plan

We create a Master Test Plan (TM 032) to manage testing activities. The Master Test Plan provides an overview of testing efforts and includes:

- Identification of broad test concepts
- Long lead resources required
- Performance criteria
- Resource constraints
- Time constraints
- Entrance and exit criteria
- Support for the six types of tests: Unit Tests, Unit Integration Tests, System Tests, System Verification Tests, System Performance Test, and System Qualification Tests (to be conducted by the State)
- Correlation to the System Development Life Cycle
- Testing automation

3C.12.9.4.1.2 Software Test Plan (STP)

We create a Software Test Plan (TM 033), in accordance with Section E2.2.2 of the IEEE J-STD-016-1995. This plan describes:

- Software test environment
- Tests planned and general information about each unit test, unit integration test, and system test
- Test schedules
- Requirements traceability

3C.12.9.4.2 Component-Based Testing Techniques

Component-based testing differs from procedural testing. Specifically, testing components requires accounting for development techniques like information hiding, encapsulation, and reuse. The alliance has years of relevant experience delivering large, component-based systems. As such, we understand the special planning considerations that component technology requires.

Components are “black boxes.” The technique of information hiding keeps other code from having to know the details of a how a component was implemented. It also makes testing more of a challenge. Users of a component cannot gain direct access to functions or procedures on modules that are part of this component. While information hiding prevents defects from propagating, it can also complicate testing. Automated unit tests cannot get access to hidden methods.

Information hiding complicates “white box” testing.

We have proven approaches to address this problem. For example, we can create a subclass of the "class under test" that exposes the otherwise private/protected methods.[69] This opens up the class for testing purposes, and testing purposes only.

In this way, we preserve encapsulation in *production* code, because we do not expose methods in production that should really be hidden. But we can still test the private methods with automated unit testing and other techniques, by using the new, testing-only subclass.

These component-based testing techniques are only applicable when writing test scripts that need access to the component's source code. For example, they are not relevant when users test an HTML page, as the underlying code is not accessible to the user for testing. Therefore, the alliance employs component-based testing techniques when unit and unit integration testing Java based components. Component testing is relevant to the Unit and Unit Integration Test levels.

3C.12.9.4.3 Multiple Levels of Testing

We employ multiple levels of testing for thoroughness. These levels include:

- Unit testing
- Unit integration testing
- System testing

The following paragraphs explain these types of tests.

3C.12.9.4.3.1 Standardized Unit Testing

The first stage of testing that we conduct is unit testing. We base unit testing activities upon clearly defined procedures, leveraging proven practices from other child support and component-based projects.

Our unit tests verify correctness of each software component's code as specified in its detailed design. Coders write, execute, and analyze the results of unit tests. This is because coders have detailed knowledge of the internal structure and logic of the code. They focus on exercising different decision statements, or paths. They also test methods with different inputs, like at and below minimums, at and below maximums, and at intermediate values.

[69] Java further complicates this with strict rules about exposing methods that have been marked "private" or "protected" using the Java language. We will work around this with wrapping methods – i.e. creating a new, "public" method that calls the internal "protected" method.

We create a unit test plan and update it as necessary to provide for repeatable tests. This plan includes:

- Documented, detailed component test conditions and expected results
- Unit test cycle control sheet

Finally, we document results of unit tests and make them available for State review.

Unit testing is typically done in an ad hoc fashion. Some developers use throwaway Java test harnesses. Others hand-write expected results and manually compare them with the actual results. We add value by standardizing the testing approach, putting it into code. We use JUnit, a popular test engine for Java. For testing Java-based web applications, we use Cactus, an extension of JUnit. JUnit provides a framework for coding unit tests and also automates their execution. This gives us a suite of regression unit tests that survives into maintenance.

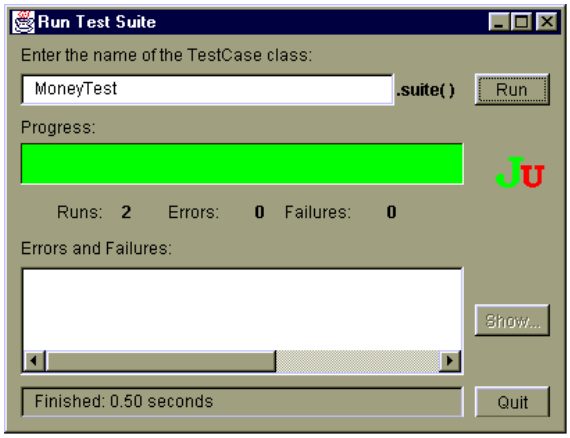


Figure 3C.12.9-4 JUnit - *JUnit aggregates unit tests into a coherent suite of tests. An easy-to-read report highlights problems and their exact location.*

Because we embed tests in Java code, they are more likely to be maintained along with code changes and enhancements. Embedding the tests also provides direct traceability (literally) from unit tests to Java code. The impact of changes to Java classes upon Java unit tests are immediately detectable using impact analysis tools built into the Java environment.

JUnit tests, embedded directly in the code, are likely to be maintained.

We code both batch and online modules in Java, so both use JUnit. See Section 3C.12.7 Software Coding for an explanation of batch and Java.

As discussed in Section 3C.12.2 System Architecture Design, the alliance uses a layered architecture, with a thin presentation layer. In other words, user interfaces –

HTML pages – do not include much processing logic. These pages depend on underlying Java logic for processing. Because they require this integration of other units, we first start testing HTML pages during Unit Integration Test. We still perform comprehensive testing; we simply recognize that that testing is better accomplished after we have tested the Java units underneath. We use Mercury Interactive WinRunner for testing HTML pages.

We unit test AI modules for message formatting, transformation, and validation using the same fundamental concepts described above. However, JUnit does not apply. Instead, we use:

- eAI tool support (to the extent the eAI bus includes such support by the time our testing starts)
- Testing tool support (for example, LoadRunner's MQ Adapter to simulate transactions)
- Custom-built test harnesses
- Traditional, hand-crafted tests where appropriate

3C.12.9.4.3.2 Unit Integration Testing

The Unit Integration Test stage begins as soon as groups of related classes and components pass unit test. It checks the interfaces *between* components, including input/output (I/O), database updates, and navigation. These procedures follow the relevant sub-clauses of IEEE/EIA 12207.2 1997 clause 5.3.8.

Unit Integration Testing has several objectives:

- Tests that classes and components work according to specification when combined into an application
- Verifies that the component interfaces have appropriately implemented the design
- Tests that transactions, database updates, and conversation flows function to specification
- Checks that the protocols defined at the design stage are respected by the implementation

The focus is now on technical integration. We have already discovered problems within individual components during unit testing. We progressively test each assembly of the application before moving to System Test. This verifies that the application works as designed.

Unit integration tests check component interactions – unit tests have already checked the internals of those components.

We conduct the Unit Integration Test according to the Unit Integration Test Plan. We update this plan as necessary to provide for repeatable tests. The Unit Integration Test Plan consists of:

- Documented, detailed Unit Integration Test Conditions and Expected Results

- Unit Integration Test Cycle Control Sheet

Finally, we document results of these tests and publish them for State review.

3C.12.9.4.3.3 System Testing

We perform System Testing upon the completion of unit integration testing for a capability group. A capability group represents a collection of related application services (e.g., business services, control services, and support services) that support a broad functional area of the system. Case management and financials are example capabilities. This test demonstrates to the State that the software meets specified requirements, in accordance with Section 3.2.28 of IEEE J-STD-016-1995. Previously completed application services within the build cycle are regression tested.

We create and update as necessary the following work products:

- System Test Description. We test the software in accordance with this description.
- System Test Report. We document the results of the System Test here.

Prior to proceeding to the next formal stage of testing, System Verification Test, we conduct a dry run of the System Test Description procedures. We check the procedures for rigor and accuracy. We also check that the system itself is ready. Based on the results of the dry run, we update the test cases and procedures as appropriate.

System Testing uses the following tools, described elsewhere in this document:

- Mercury Interactive Test Director, for managing test plans, test cases, and recording defects
- Mercury Interactive WinRunner / Astra QuickTest for automated functional test scripts

In addition to these functional tests, we also conduct performance testing with Mercury Interactive LoadRunner. This tool simulates load and measures throughput, response time, and availability. The Mercury Interactive Suite is discussed elsewhere in this document.

3C.12.9.4.3.4 System Verification Testing

System Verification Testing is conducted after we complete System Test. It demonstrates that the system, as a whole, correctly satisfies the functional requirements.

We include performance and regression testing in System Verification Test.

System Verification Test confirms that the system, as a whole, satisfies functional requirements.

We conduct a System Verification Test Readiness Review prior to test execution to confirm the system test description will validate requirements. As part of System Verification Test, we produce:

- System Verification Test Description (TM 034-2). We test the software in accordance with this description.
- System Verification Test Report (TM 035). We document testing results in this report.

System Verification Test continues to use the Mercury Interactive testing tools.

3C.12.9.5 Thorough and Accurate Test Data

This section covers the approach to obtaining, preparing, and maintaining test data, including the identification of test data sources. Our experience shows that preparing thorough and accurate data requires:

Centralized management removes redundancy and inaccuracy.

- Centralized management of data
- Use of multiple data sources
- Planning for maintenance of the existing database inventory

3C.12.9.5.1 Centralized Test Data Management

Centralized test data management promotes effective reuse of test data across testing efforts, for the development, architecture, and testing teams. The following paragraphs discuss this approach.

3C.12.9.5.2 Multiple Data Sources

We obtain a variety of data from multiple, independent sources. This improves our ability to detect errors that arise only with particular test data. We obtain test data in the following ways:

- Copy data from a production database. Depending on the timing and type of data required, this data will be run through the CCSAS CSE data conversion process. This allows us to exercise the system with “real life” data.
- Create particular data cases from scratch. The testing team may either enter it using completed portions of the application or using SQL scripts.

- Generate data using automated tools. This is primarily for performance and load testing, which require large data sets. In these cases, data volume is important but content (i.e., functional details) generally is not.
- Receive interface data from external entities. The alliance works with the SDU vendor and external entities that interface with CCSAS CSE to receive data representative of the transactions the external entity will forward. During system testing, the alliance works with external entities to create interface transactions that can be processed by the recipient and returned to the sender.

We note that we always test against *test databases* rather than live production data.

3C.12.9.5.3 Planned Maintenance

We maintain the existing database inventory. We preserve copies of System Test data throughout testing and into maintenance. We do this with DB2 database backup and restore utilities. Additionally, we preserve a database of general-purpose data for use during testing. We can tailor data from this database for a specific test. In doing so, we need not repeatedly recreate basic test data needed by several tests. Such data might include common tables, people, and user profiles.

We preserve the relationship between test documentation (e.g., conditions, scripts, cycle documents) and System Test data. We do this with the process described in Section 3C.4 Configuration Management.

The alliance updates preserved data as changes or enhancements impact the data model. The Software Test Plan addresses the method for updating and synchronizing this test data. Lastly, we make the same test data available across the various execution environments of the application.

3C.12.9.6 Interfaces – External and SDU

We test interfaces using the same methodology for testing other software components.

Testing requires a high level of coordination, communication and cooperation among project personnel. This includes the development team, interfacing agency technical staff, interfacing agency management, and CCSAS programming and technical staff. We work together to verify test data, transactions, schedules, results, and resolve discrepancies where required. The team provides test files in the format that agencies use in testing their receipt of CSE interface data.

To facilitate round trip testing during system testing, the alliance works with external entities to create interface transactions that can be processed by the recipient and returned to the sender.

We use XML where possible for system interfaces, as noted in Section 3C.12.8 System Integration. XML is human readable and contains meaningful tags to delimit data. This contrasts with traditional fixed-length records, which are often error-prone and more difficult to verify. Our use of XML therefore expedites testing activities for interfaces, by providing a readable and unambiguous view of the data to verify test results.

3C.12.9.7 Regression and Performance Testing

This section describes how we incorporate regression and performance testing into testing activities. It also includes our approach, methods, and tools for regression and performance testing.

3C.12.9.7.1 Performance Testing

We use a performance pilot, as necessary, for early performance testing. We base the pilot on performance requirement specifications defined by the State. Later, we incorporate performance testing into Unit Integration Test and System Test. At the final stages of testing, we conduct performance testing in the CCSAS Performance Environment. This production-like environment is designed for high volume web serving, and it replicates the CCSAS production environment on a smaller scale. Use of an environment that closely resembles production provides realistic measurements of throughput and scalability.

3C.12.9.7.1.1 Performance Pilot

Depending on infrastructure risks, we may conduct a performance pilot early on in the development lifecycle. This would help prove out part of the system architecture design. A pilot validates that the architecture meets non-functional requirements as designed, is decomposed into components, and is layered.

Piloting is consistent with the Rational Unified Process.

3C.12.9.7.1.2 Performance Testing in Unit Integration Test

As we integrate unit-tested components, we begin a rigorous performance testing effort.

We use JProbe by Sitraka Software for performance testing. This code profiling tool identifies and reduces Java performance bottlenecks. It probes the program's calling relationships and program structure. While code executes, JProbe identifies inefficient algorithms, excessive object creation, input/output blockage, excessive method calling, and inefficient memory usage.

JProbe helps focus the fixing efforts for the most impact, adhering to the "Pareto rule of performance testing." This rule states that 20% of an application's code takes up the 80% of the time required to run the application. By using JProbe, we identify the major bottlenecks in the 20% of the code.

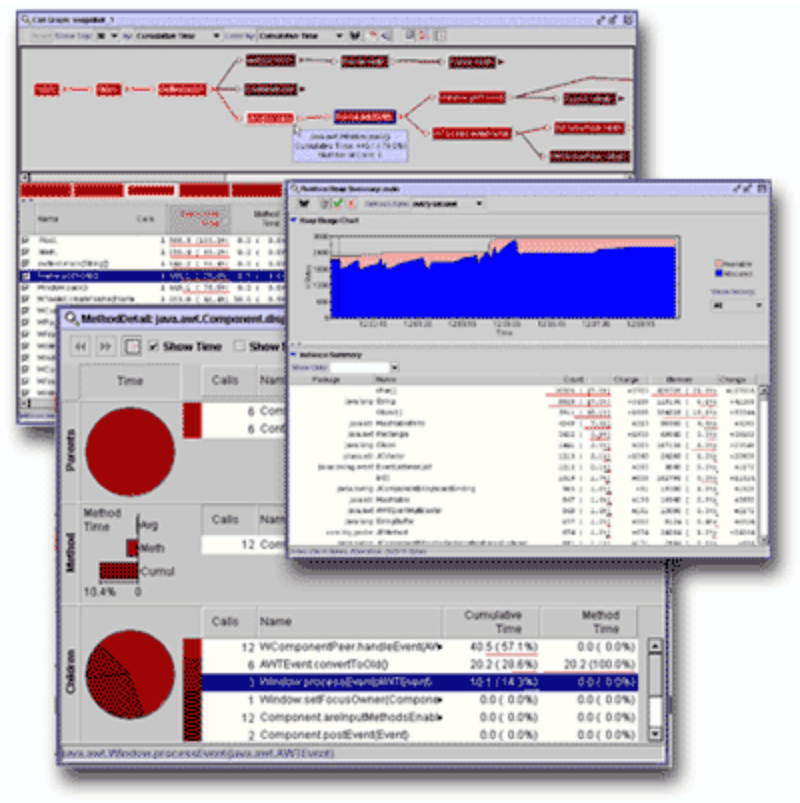


Figure 3C.12.9-5 Code Profiler - *We use a JProbe to avoid making changes aimlessly in the hopes they improve performance.*

3C.12.9.7.1.3 System Performance Testing

We develop a System Performance Test Description.

System Performance Testing simulates:

- Workstation tasks, batch processing, and interface processing
- Daily and periodic non-routine conditions
- Local office and central processing functions

We conduct test performance during System Testing. At this point, the code is fairly stable. This increases accuracy and effectiveness of the performance test above what it would have been at an earlier stage. We also conduct ongoing performance testing prior to production software releases. We leverage existing system test cases as starting point for performance test case development.

Finally, we use Mercury Interactive LoadRunner to perform stress testing. Stress testing validates that the application can scale to a large number of simultaneous users. It also validates that the application can handle large amounts of data. LoadRunner exercises an entire enterprise infrastructure by simulating thousands of users and employing performance monitors to isolate problems.

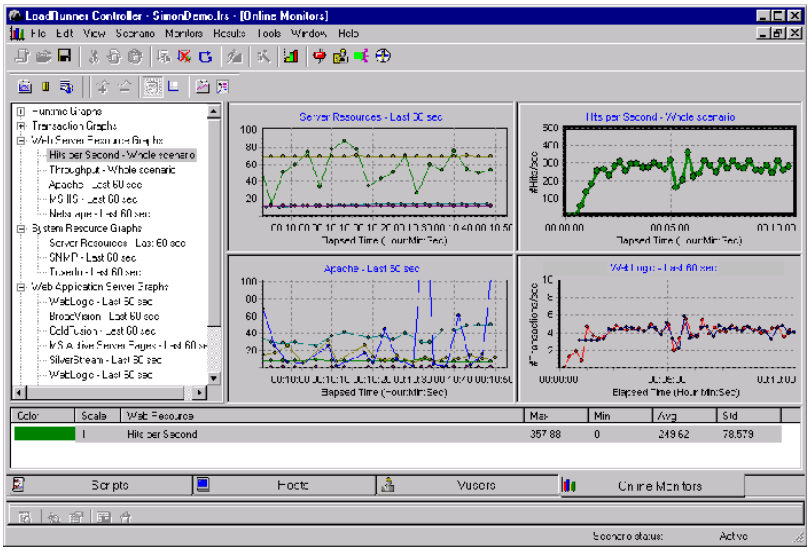


Figure 3C.12.9-6 Stress Testing Tools. *LoadRunner simulates thousands of concurrent users – matching CCSAS' usage profile (Low-resolution screen shot).*

3C.12.9.7.2 Regression Testing

Regression Testing confirms that defect resolutions do not introduce new, unintended defects. We recognize that this is particularly important with component approaches, which emphasize reuse. Making a change in one component can require re-testing the related interactions with other components. Regression testing is also valuable for complex functionality, like payment distribution. Here, the chances are greater that fixing a defect might inadvertently break something else.

We also expect to regression test complex, child support financial processes. By running large payment batches through the system, we can verify accurate accounting for financial transactions.

We are looking at ways to incorporate Regression Test into Unit Test and System Test. We already discussed the use of JUnit for regression unit tests earlier, in subsection 3C.12.9.4.2.1 of this document.

3C.12.9.7.2.1 Regression Testing Tools Automate Testing

We use Mercury Interactive's WinRunner/Astra QuickTest package to perform regression tests. This tool supports the record and playback of graphical user interface (GUI) regression tests. Users run the application and WinRunner/QuickTest records their mouse movements and typed-in data. WinRunner/QuickTest allows us to automate business events and automatically compare responses to mouse or keyboard inputs.

Regression testing is particularly important with incremental, component-based development.

3C.12.9.7.2.2 Regression Testing in System Test

We regression test previously completed capabilities when we promote a new capability to System Test. This tests that the newly promoted capability does not adversely impact other capabilities. This is particularly helpful, for example, with the early-win approach. We perform a System Test on the Statewide Services components at the end of Version 1. With regression testing, we can more easily repeat this test at the end of Version 2, when county services migrate to our next-generation architecture.

Regression testing allows us to more easily test previously delivered increments.

3C.12.9.7.2.3 Regression Testing in System Verification Tests

During System Verification Test, we regression test to confirm existing system capabilities continue to function properly, as new capabilities are rolled out under our incremental development approach.

Regression testing also confirms existing system capabilities when we need to make significant changes or enhancements to the hardware or application architecture.

3C.12.9.8 Independence in System Test

To achieve independence in System Test, we establish a separate and independent testing team. Alliance personnel managing this team are not the same individuals responsible for overseeing design and development.

Development personnel with knowledge of software items may, however, contribute to this process under the Testing Team's direction when their point expertise is needed. For example, these personnel may help with test cases that rely on their knowledge of the software's implementation. This is in accordance with Section 5.9.1 in the IEEE J-STD-016-1995.

3C.12.9.9 Input into Build Planning Activities

We incorporate the outcome of testing into software build planning in a variety of ways:

- We compare test outcomes against defined exit criteria. This proactively mitigates risk for moving onto planned activities that depend on test outcomes. In other words, we practice stage containment.
- We use test outcomes to produce test metrics. The System Test Report and the System Verification Test Report (TM035) are used in the determination of operational readiness. These reports include metrics that help validate test activities to plan, identify deviations, assess the impact on other build activities, and adjust build plans accordingly.
- Our system development approach is incremental. We develop portions of a capability (i.e., an increment) from start to finish. Then, we can demonstrate some of the application earlier in the project instead of waiting until completing all development. As such, we apply lessons learned from test outcomes to the development approach going forward. We "build a little; test a little." We incorporate this learning into the plans for constructing future software components.

Software build activities during testing follow the build process described in Section 3C.4 Configuration Management.

3C.12.9.10 Input into Problem Resolution Process

We record problems encountered by the testing team while executing unit integration and system testing. We follow the problem resolution process described in Section 3C.10 Problem Resolution. We store and manage these problems in a problem resolution repository. This initiates our formal problem resolution process.

The development team lead and testing team lead jointly review problems frequently. They review the status of open items and confirm and prioritize new problems. We document, maintain, monitor, and communicate status of problems discovered while testing until we confirm their resolution through retesting.

3C.12.9.11 System Qualification Test Support

The State plans, executes, and evaluates the results of State-run System Qualification Testing. We support the State in these activities with training, planning assistance, and functional and technical support:

- We conduct training to prepare LCSA and State DCSS staff for System Qualification Testing. This training provides an overview of CSE functionality.
- We provide functional and technical support during State personnel execution of the system qualification test. Technical support includes, for example, loading test data sets into the test environment and running batch scripts. Functional and technical support comprises a limited number of FTE's for a specified number of hours, to control scope.
- We help the State interpret the final runs of System Qualification Testing.

3C.12.9.12 Conclusion

Software defects happen. Large-scale systems like CCSAS CSE require a robust testing process to detect and resolve these defects. Moreover, it is important to begin testing *early* in the system development lifecycle, to contain costs spent on fixing defects that would otherwise propagate.

We bring a disciplined and structured approach to testing. This includes early, up-front planning; proven techniques for stage containment; and regression testing and automation where possible. We focus on testing thoroughness and rigor: creating and maintaining plans, producing documentation, incorporating multiple reviews and checkpoints throughout the process, and evaluating our progress with metrics. We conduct System Verification tests to validate that we have satisfied the State's requirements. Finally, after completing system testing, we continue to focus on testing by supporting the State in its own System Qualification Testing efforts.

Our robust testing process detects problems, early on, and tracks them with a formal problem resolution process.

3C.13.0 System Implementation

Our proposed system implementation approach combines techniques proven successful in California and adds best practices from our other projects. To execute our solution, the alliance brings the combined expertise of the nation's largest and most successful implementers of CSE systems together with local vendors who possess deep functional and technical knowledge of all six California CSE consortia systems.

3C.13.0.1 Introduction

As stated in the CCSAS CSE Solicitation for Conceptual Proposal: "Implementation of the new system is constrained by disparate systems, complex business rules, number of locations, and changes in the way of doing business." We understand the significance of this statement based on our considerable and direct experience in California and other states.

The California child support program is in the midst of considerable change that will impact the CCSAS project. Since 1998, the State has been actively transitioning counties to one of its six safe-haven consortia systems. By early 2002, the 58 counties are scheduled to operate on a State-approved consortia system, effectively reducing the number of systems in operation from approximately 25 down to six. While this consolidation of county systems represents a significant achievement, the remaining systems do differ, and in some cases substantially, in their database constructs, business rules, level of automated functionality, and hardware/software environments.

Change also has been occurring on the program front. The passage of AB196 in September of 1999 created a new Department of Child Support Services (DCSS) and moved responsibility from local service delivery by county district attorneys to statewide service delivery by the DCSS. This shift in program responsibility is well under way with the final county transitions scheduled to be completed by July of 2002. Also noteworthy is the fact that the DCSS has examined child support business best practices and is targeting the standardization of program operations throughout the State. Doing so will create uniformity of service from LCSA to LCSA, a benefit to the customers of the child support program. The CCSAS CSE project presents an opportunity to contribute to these efforts by placing California CSE staff on one system that is designed and implemented with uniform business processes as a foundation.

In response to the dynamic nature of California's child support program, the alliance has designed a two-phased system implementation solution focused on mitigating risks that

might otherwise impede the successful implementation of a statewide system. Specifically, our proposed solution leverages four assets the alliance brings to this project:

- A baseline implementation methodology centered on our experience of teaming with the DCSS to successfully transition over 35 California counties to their safe-haven system of choice
- Lessons learned from the statewide implementation of the Child Welfare Services Case Management System (CWS/CMS)
- Experience from the combined expertise of the nation's largest and most successful implementers of child support systems
- An implementation team with deep technical and functional knowledge of the six consortia systems

Our implementation solution encompasses four main areas, which contain seven disciplines, or specialized areas necessary to complete implementation efforts. The sections that follow provide more detail on these efforts:

- 3C.13.1 Data Conversion
- 3C.13.2 Transition Management, which includes Change Management, Implementation Management, On-Site Support, and Local Interfaces
- 3C.13.3 User Training
- 3C.13.4 Hardware/Software Installation

This section, 3C.13.0, provides a framework for our implementation solution. It covers:

- Leveraging alliance assets
- Implementation approach
- Understanding the implementation approach under the two phases of the CCSAS CSE project
-

3C.13.0.1.1 Leveraging Alliance Assets

Proven Implementation Methodology

For the past several years, our alliance member Synergy has actively worked with the DCSS to manage the transition of over 35 counties from their legacy applications to one of California's six consortia systems. In early 1998, Synergy, in conjunction with the Health and Human Services Data Center (HHSDC; it was called the Health and Welfare Data Center at that time), developed an implementation framework used to guide the transition of 17 counties to the CASES consortium. The success from that effort led to the adoption of the implementation framework for use on future county transitions managed by the State. Over time, our system implementation approach matured as lessons learned from prior county transitions were incorporated. Today, this same approach to county transitions serves as the cornerstone for our proposed system implementation solution for the CCSAS CSE project.

Lessons Learned From CWS/CMS

To further solidify our baseline solution, the alliance also applies lessons learned from IBM's experience on the Child Welfare Services Case Management System (CWS/CMS) project. Much like child support, the child welfare program in California is state operated and county-administered. The CWS/CMS project culminated in the rollout of a new application to the 58 California counties and today remains the State's only large-scale, statewide human services system. This experience, unique to the alliance, provides us with keen insight into statewide system implementations. A few of the more notable lessons include:

- A successful conversion process included iterative reviews and cleansing of county data prior to final cutover to CWS/CMS. This led to a significant reduction in errors, resulting in a smoother transition. This iterative review and cleansing process is part of the alliance conversion management process.
- The successful rollout of the CWS/CMS application was enhanced by having Implementation Consultants participate with multiple counties such that lessons learned in one county were used in subsequent counties to enhance cutover, and avoid common pitfalls such as providing training too soon to the users.
- Assisting counties in understanding their future business processes based on the CWS/CMS application was very important. Counties that dedicated themselves to identifying and making the necessary changes achieved a higher level or user participation and satisfaction. The alliance outreach, orientation and business process activities are directed at increasing county participation.
- Traditional education focusing specifically on the application may not provide users with a complete understanding of how a system can be integrated into their daily routine. The alliance's user training approach uses scenarios to instruct users in how the CCSAS CSE application can help to organize and manage work.
- The ability to manage technology refreshes for both hardware and operating system software was enhanced during CWS/CMS. Based on the CWS/CMS lessons, the alliance understands the implications associated with successfully rolling out a refresh of over 12,000 desktop hardware and operating system upgrades and the importance of the desktop to support new releases of the CCSAS CSE application.

National Implementation Experience

The alliance's approach applies the combined experience of the nation's largest and most successful implementers of child support systems. To develop a tailored CCSAS implementation approach, we considered alliance projects similar to CCSAS CSE in the following ways:

- Experience on statewide child support projects that demonstrate familiarity with the user population and system functions, similar scope in terms of business processes, and knowledge of statewide implementation efforts

- Experience with projects in county-administered environments that demonstrate an understanding of the complexities in working with county organizations having differences in demographics
- Experience with California state agencies and the political and organizational environment
- Experience with other human services projects that demonstrate familiarity with the interfacing agencies and similar user populations
- Experience with browser-based systems that demonstrate the different factors in implementing a thin client solution versus a mainframe or client-server system

Consortia System Expertise

As pointed out in the SCP, the fact that California currently operates six disparate systems adds complexity to the overall implementation effort. The alliance recognized early on that expertise with each consortia system would be a valuable, if not essential, component of a comprehensive implementation solution. Accordingly, the alliance sought out and brought together the only multi-vendor team that can provide end-to-end expertise of California's consortia systems. As Figure 3C.13.0-1 demonstrates, our team is unmatched in its ability to address the specific characteristics of each consortia system.

Consortia System	Alliance Partner
CASES	Informatix
ARS	Endeavor
KIDZ	Synergy/Accenture
STAR/KIDS	AMS
BEST	CSE Resources
CHASER	Ken Cushman

Figure 3C.13.0-1 Alliance Partners - *The alliance team brings experience with the six California consortia systems, an unmatched qualification.*

3C.13.0.1.2 System Implementation Goals

As we set out to architect our specific implementation approach for the CCSAS CSE project, the alliance focused on achieving four key goals. Specifically, the alliance's implementation solution is aligned to:

- Reduce LCSA disruption
- Mitigate risk
- Adopt statewide uniformity
- Promote a strong partnership with the State and counties
-

Reduce LCSA Disruption

Inherent in large, complex system implementation efforts is the challenge to alleviate disruption. For California's child support program, excessive disruption to LCSA operations could lead to decreased levels of customer service and potential degradation of statewide performance metrics – two outcomes that must be avoided. While eliminating disruption entirely is not practical given the scope of the CCSAS CSE project, our approach incorporates efforts to reduce its effect. We fully recognize the need to continue to meet time-sensitive federal activity and reporting requirements and the needs of the families, caseworkers, employers and other stakeholders during the implementation period.
Reducing disruption in the LCSAs can be viewed from two dimensions: pre-cutover and post-cutover. Pre-cutover, our solution places particular emphasis on limiting disruptions on day-to-day county operations by carefully coordinating county "touch points," or key events, throughout the implementation planning period in each county. We will work closely with LCSA management early in the implementation process to identify needed resources and their anticipated level of involvement. We realize the importance of setting expectations early on so that LCSA management can plan effectively for participation of their staff.

In addition, our implementation approach includes several innovations designed to keep LCSA staff focused on program delivery during the pre-cutover phase:

- Each LCSA implementation is managed by a dedicated Implementation Manager, who serves as the single point of contact for transition activities in that county. Our goal is to streamline communications and avoid overlapping or uncoordinated requests of county staff, and to provide a single accountable party. The Implementation Manager has tools available to support him/her in efforts to coordinate readiness for the county, including the Readiness Metrics included in the Pre-Implementation Readiness Assessment (TM 038) and the Implementation Planner, a workbook to guide the LCSAs in completing tasks. The role of the Implementation Manager and the process used to implement LCSAs is described further in Section 3C.13.2.3.
- The user training methodology described in Section 3C.13.3 employs a coach based e-learning approach that reduces the time LCSA staff are away from the office. Our approach utilizes highly effective self-paced training courses that can be taken at the user's workplace, supported by an on-site coach who is available for one-on-one guidance.
- The data conversion process includes two tasks that typically require a high level of LCSA involvement: data mapping and data cleansing. Because the alliance can provide staff with deep knowledge of the six consortia system data constructs and rules, county involvement in the data mapping effort will be reduced. Furthermore, our conversion approach, described in Section 3C.13.1, relies on an automated data cleansing tool rather than paper reports to screen rejected data. As a result, LCSA staff can more efficiently review, prioritize, and cleanse data.
- Our proposed county-based hardware/software installation strategy, described in Section 3C.13.4, calls for equipment to be installed and tested during evening rather than business hours to reduce impact on LCSA operations.

Post-cutover, our implementation solution focuses on providing support to users with the goal of having a smooth transition with negligible disruption to daily LCSA activities. Recognizing this goal, we script the cutover process and schedule conversion activities over the weekend to alleviate impact during business days. We also provide a period of on-site support to users during the critical time following transition, with project staff working hand-in-hand with users to allow them to continue their daily work using the new system (described further in Section 3C.13.2.5). Following this initial period of on-site support, the first point of contact for user support transitions to the Help Desk.

Mitigate Risk

As with any implementation effort of this magnitude, mitigating risk is a top priority. Knowing this, we place a premium on identifying these risks up front and developing potential mitigation strategies for each.

For example, our implementation solution recognizes the complex nature of the CCSAS CSE project and the need for close coordination between DCSS, CCSAS project staff, and LCSAs. To help facilitate this coordination, we propose having regional meetings on a regular basis with representatives of these agencies in attendance, with the content of the meetings focused on the CCSAS CSE project. The objectives and proposed structure for these meetings are described further in Section 3C.13.2.3.

Adopt Statewide Uniformity

As part of its system implementation approach, the alliance participates in State child support business standardization activities as necessary and appropriate to facilitate uniformity and successful system transition and implementation. Our participation includes identification and reporting of standardization issues and review of business practices, policies, procedures, and regulations. We fully understand the importance of our role to support the DCSS to provide high quality services in a manner that is responsive to the needs of its customers. Successful development of CCSAS will go a long way toward enabling delivery of high quality service.

We understand the DCSS management's commitment to leading California to statewide uniformity concurrent with the development of the CCSAS CSE system. It is clearly reflected in the child support reform statutes that the current program seeks to avoid the considerable disparity in day-to-day operations from county to county. LCSAs must provide consistent, effective service in the child support program. There must be uniformity of philosophy, which drives the approach or methods used to establish and enforce support orders, resource allocation, and program policy, procedure and practice. It is under the leadership of the CCSAS project and DCSS, with user involvement and support from the alliance, that CCSAS will bring uniformity to the way that counties provide child support services through a single statewide application and environment that includes uniform desktops, tools and services.

Promote a Strong Partnership with the State and LCSAs

The alliance implementation methodology also utilizes a joint team concept, proven to be effective at easing the effect of change on users and transferring valuable knowledge to those who will eventually become responsible for the ongoing use, maintenance, and operation of the system and associated procedures. We encourage the active participation of the CCSAS project staff, DCSS, and LCSAs to plan and execute the specific elements of the implementation methodology and approach. In doing so, State and county staff are involved in the planning, have ownership of the result, and gain the hands-on experience they will need to effectively use, operate, and maintain the system, which they have created and implemented.

A complex implementation such as CCSAS CSE requires an implementation approach that is comprehensive, repeatable, and flexible.

3C.13.0.2 Implementation Approach

The alliance CCSAS CSE implementation approach focuses on up-front planning to result in an efficient transition in the implementation counties. This approach has proven to provide a useful framework that increases the efficiency and quality assurance features of a standard methodology, yet also provides opportunities to customize the approach based on specific LCSA characteristics.

Managing System Implementation

The alliance approach to implementation tackles these challenges through the integration and use of structured, proven project management policies, standards and procedures that will be established for all aspects of the CCSAS project. By using an integrated approach to project management, the implementation organization will have ready access to the same tools, methods, and procedures used by the Project Management Office for overall management of the project. This will hasten the identification, resolution, and mitigation of issues and risks, enabling the project's overall success.

Implementation Approach

The alliance has developed an implementation approach that is ideally suited to the challenges of managing a complex system implementation in a diverse county-administered environment.

The alliance implementation methodology, as shown in Figure 3C.13.0-2, is composed of seven implementation disciplines that work to ready a county for transition across four main implementation steps. The seven implementation disciplines are introduced below. The four steps are Start-Up, Readiness, Pre-Cutover, and Post-Cutover; these steps are described in more detail in Section 3C.13.2.3.

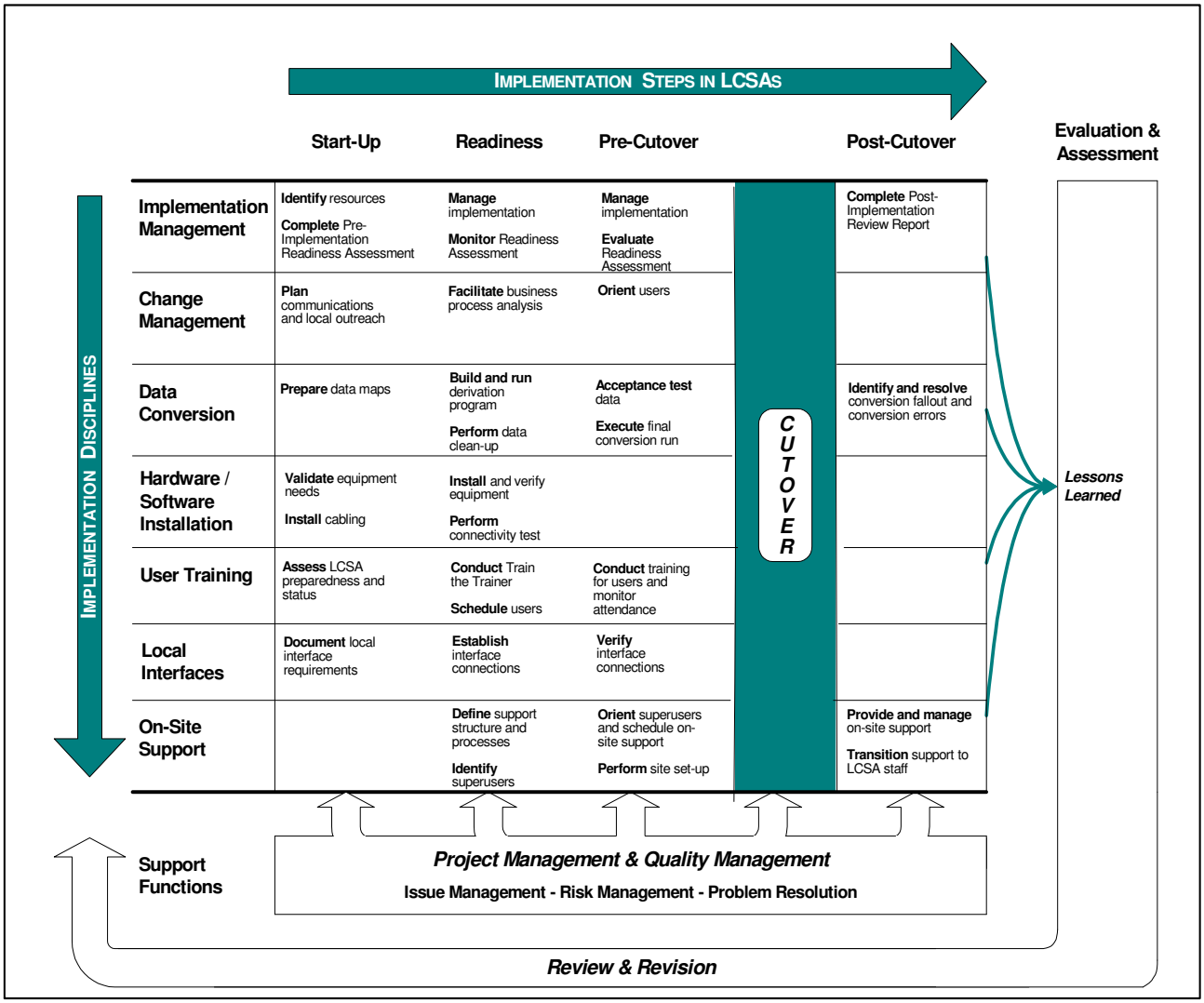


Figure 3C.13.0-2 Overview of Implementation Approach by Discipline and Steps - *The seven implementation disciplines work throughout the four implementation steps to prepare each county for transition.*

Seven Implementation Disciplines

The alliance implementation approach introduces seven key disciplines that encompass the major implementation activities to be performed on the CCSAS CSE project. Taken together, these disciplines cover the four key readiness components of a system implementation: data readiness, user readiness, organizational readiness, and technical readiness.

The specific definition and areas of responsibility associated with the seven disciplines as addressed by the alliance's implementation approach are described in the following table:

Implementation Disciplines

- Site Implementation Management –Coordinates activities necessary to implement a new system in a site, including managing the key relationships with the LCSA Director and LSCA Coordinator, and managing the efforts of the alliance staff working in the county. Activities from conversion to training to on-site support are a part of the overall Implementation Management coordination and monitoring responsibilities.
- Change Management – Provides outreach and orientation communications about the project to align expectations and manage the impact of change on users and external entities. Also includes business process analysis work to promote uniform processes.
- Data Conversion – Develops a conversion strategy for each data source requiring conversion and performs data conversion.
- Hardware and Software Installation – Installs the hardware, software, and cabling for sites prior to cutover, including installation and verification activities.
- User Training – Develops and delivers training to users, including developing e-learning modules and installing a Learning Management System.
- Local Interfaces – Establishes connections and validates county-specific interfaces, such as the TANF and county court interfaces.
- On-Site Support – Performs site set-up activities to allow necessary components for system implementation to be in place and operational prior to cutover. Provides support to users after cutover to allow a smooth transition and reduce disruption.

Figure 3C.13.0-3 The Seven Implementation Disciplines - *Each implementation discipline covers specific activities associated with providing a comprehensive implementation effort.*

3C.13.0.3 Understanding the Implementation Approach Under the Two Phases of the CCSAS CSE Project

The alliance proposes to use the same general implementation approach in both phases of the CCSAS CSE project. The approach is designed to be flexible to accommodate any specific differences that occur by phase. The main implementation differences occur in the division of responsibilities and some of the specific tasks to be conducted for each phase. We have proposed a division of responsibility between CCSAS project, LCSA, and the alliance primarily based on the expertise with the various systems being

implemented. The following sections describe how our approach will be applied under a two-phased project.

The primary objectives of Phase I are to:

- Implement the statewide services component of CCSAS CSE, including the Statewide Case Registry (SCR), extended locate services, federal reporting, SDU interface, and limited CSENet interface functions.
- Transition to a two-consortia environment, by converting the KIDZ and STAR/KIDS consortia LCSA to the CASES consortia system.
- Enable the State of California to begin realizing benefits and increase program performance early in the project lifecycle.

The primary objectives of Phase II are to:

- Implement the federally certifiable Version 2 statewide child support enforcement system in all LCSAs andstate DCSS.
- Improve the quality, consistency and timeliness of the delivery of services to recipients of CSE program services.
- Reduce the complexity, cost, and time required to operate, manage and maintain the CSE system, interfaces and technical infrastructure.
- Enable the State of California to realize benefits in the CSE program through increased collections, improved customer service, reduced cost of maintenance and operation, and increased worker effectiveness.
- Construct the LCSA-level functionality of the Version 2 solution, which when integrated with Version 1 provides the State with CCSAS CSE, a single statewide system

3C.13.0.3.1 Understanding Implementation – Phase I

Phase I is designed to: a) reduce the number of consortia systems from six to two b) implement the Version 1 of our CSE solution. More specifically, Version 1 will serve to link two of the existing consortia systems, ARS and CASES, to provide the functionality necessary to create a statewide case registry, improve locate, and implement statewide reporting.

The alliance strategy for Phase I implementation leverages the current methods and delivery approach that have been used during the CASES transitions to date. In order to divide the responsibilities of the implementation activities, we explored the current CASES approach and placed responsibility for implementation activities with the organizations that had the most experience conducting those activities. For example, to provide a coordinated approach, particularly as the Phase I effort entails a more compressed timeline than has been underway to date, the alliance will perform management and coordination activities.

Our Phase I strategy attempts to reduce the overall system implementation scope where possible. By re-using much of the existing CASES infrastructure that has already worked well, our solution reduces implementation risk and the potential for county and program disruption as we deploy Phase I of the CCSAS CSE project.

The following key assumptions are important to understanding the scope of the Phase I implementation:

- 14 non-CASES, non-ARS counties transition to the CASES consortium.
- 55 future CASES counties run at the Health and Human Services Data Center (HHSDC). The 34 existing CASES counties either currently operate out of HHSDC or are in the process of transitioning their operations to HHSDC.
- CSE Version 1 operates out of the new CCSAS CSE Data Center.
- The HHSDC WAN provides connectivity between the 55 future CASES counties and HHSDC.
- Existing CASES customer service and help desk capabilities continue to be used (though are scaled up by the State to accommodate the increase in caseload and user population).
- The ARS consortium, composed of Los Angeles, San Diego, and Orange counties, continues operating under the current status quo.

3C.13.0.3.2 Understanding Implementation – Phase II

During Phase II, the alliance finishes design and construction of the Version 2 solution and rolls it out to 58 counties. The current ARS and CASES systems are retired, thereby completing the implementation of a single, statewide automated system. During Phase II, the alliance works with the DCSS and other State departments to implement other key aspects of our solution including an integrated Central Case Registry, a Service Center, a statewide Interactive Voice Response component (IRV), and the CCSAS CSE project website. Other critical state-level business functions such as tax intercepts and financial institute data match also are incorporated within Version 2.
As the alliance will have the most experience with the new CSE Version 2 system, we take on the bulk of the implementation activities for Phase II, in coordination with the LCSA staff whose involvement and buy-in is necessary for a successful implementation.
The following key assumptions are important to understanding the scope of the Phase II implementation:

- 58 counties transition to Version 2 of the CSE system. County transitions will therefore be focused on moving away from two systems: ARS and CASES.
- Operations are centralized at a new CCSAS CSE Data Center. In other words, county operations are added to the Version 1 operations established during Phase I.

Figure 3C.13.0-4 presents a summary of the division of implementation responsibilities by phase.

Phase I Implementation: Division of Primary Responsibilities			**Phase II Implementation: Division of Primary Responsibilities**	
• Management and Coordination	• alliance	**Implementation Management**	• Management and Coordination	• alliance
• Pre-Implementation Readiness Assessment	• alliance		• Pre-Implementation Readiness Assessment	• alliance
• Post-Implementation Review Report	• alliance		• Post-Implementation Review Report	• alliance
• Management and Coordination	• alliance	**Change Management**	• Management and Coordination	• alliance
• Statewide Outreach	• State		• Statewide Outreach	• State
• Local Outreach	• State		• Local Outreach	• State
• Orientation	• State with alliance support		• Orientation	• State with alliance support
• System Functionality Maps	• Alliance		• Business Process Analysis	• Alliance
• Management and Coordination	• alliance	**Data Conversion**	• Management and Coordination	• alliance
• Data Mapping and Derivation	• alliance		• Data Mapping and Derivation	• alliance
• Data Validation and Cleansing	• alliance		• Data Validation and Cleansing	• alliance
• Data Acceptance Testing	• State		• Data Acceptance Testing	• alliance
• Management and Coordination	• alliance	**Hardware/ Software Installation**	• Management and Coordination	• alliance
• Local System Setup	• alliance		• Local LCSA Setup	• alliance
• State-Level System Setup	• State		• State-Level System Setup	• alliance
• WAN Connectivity	• State		• WAN Connectivity	• State

Phase I Implementation: Division of Primary Responsibilities			**Phase II Implementation: Division of Primary Responsibilities**	
• Management and Coordination • Materials • Delivery • Infrastructure (Enviroment)	• alliance • State • State • alliance	**User Training**	• Management and Coordination • Materials • Delivery • Infrastructure	• alliance • alliance • alliance • alliance
• Management and Coordination • Connection to IV-A and County Auditor/Controller Interfaces	• alliance • alliance	**Local Interfaces**	• Management and Coordination • Connection to IV-A and County Court Interfaces	• alliance • alliance
• Management and Coordination • Site Set-Up • On-Site Support	• alliance • alliance • State	**On-Site Support**	• Management and Coordination • Site Set-Up • On-Site Support	• alliance • alliance • alliance

Figure 3C.13.0-4 Summary of Implementation Responsibilities by Phase - *The primary responsibility for delivering an implementation task resides with the party with the experience most relevant to that task.*

3C.13.0.4 Conclusion

Our direct involvement in California's child support program for the past seven years gives us a firm understanding of the technical, programmatic, federal, and political issues that face the State. The alliance also is aware of the challenges posed by implementing a new system in an environment of complex program rules, procedures, practices, and distinctly different current automation support and worker automation expertise. In response, we are proposing a comprehensive system implementation solution based on a proven framework that has delivered over 35 California counties to their safe haven system. To that foundation we have added lessons learned from the CWS/CMS statewide implementation, and best practices and tools from our experiences on other projects. Delivering our solution is a team with deep system implementation expertise, and of critical importance, direct experience with the six California CSE consortia systems. When taken together, our implementation solution is tailored to the challenges that face California.

3C.13.1 Data Conversion

The alliance is uniquely positioned to propose a conversion approach that brings together the same methodology, automated tools, and consulting expertise successfully used by the State to transition over 35 California counties to their safe-haven system of choice.

3C.13.1.1 Overview

Data conversion is perhaps the single most significant implementation challenge California faces in the transition to a single, statewide child support system. As such, the alliance places considerable emphasis on mitigating risk through the re-use of a conversion approach that boasts a solid, long-standing track record within California.

Since early 1998, the State has actively supported the transition of over 35 counties from numerous legacy CSE applications to one of six safe-haven systems. The vast majority of these counties relied on a data conversion methodology and automated toolset developed by the alliance. Over time, this framework has been carefully refined and today continues to provide the State with a mature, robust solution for transitioning the few remaining counties to safe-haven systems.

Our conversion methodology is based on several guiding principles:

The complex data sets inherent in child support systems must be divided into discrete phases to simplify the end-to-end conversion process.
The conversion scope is significant with over 2,000,000 IV-D cases from different consortia systems and multiple State-level repositories.
Data conversion must be done iteratively so that data is properly mapped, edited, and thoroughly cleansed prior to cutover.
Conversion edits and validations applied at cutover must be standardized for consistency across converting counties.
Data mapping and derivation must be flexible to support differing LCSA legacy sources.
Successful data conversion requires a coordinated effort between State and alliance staff.

Our conversion methodology is supported by a toolkit that provides efficiencies throughout the conversion cycle. At the heart is the Automated Conversion System (ACS), a powerful tool for performing data edits and validations on data bound for the target system. In effect, the ACS serves as the “conversion bridge” into these systems

by only allowing cleansed data to cross. To date, the alliance has successfully built and utilized an ACS for the ARS, CASES, KIDZ, and STAR/KIDS consortia.

Also included in our toolkit are the Automated Data Dictionary and Data Mapping Tool (ADD/DMT) and the Automated Data Cleansing Tool (ADCT). These tools support documentation of conversion requirements, capture data mapping rules/decisions in a single repository, and efficiently facilitate the tedious process of data cleansing.

Our proposed Conversion Methodology and Toolset (CMAT) serves as the catalyst to drive LCSA conversions on the CCSAS CSE project with one more critical ingredient: knowledge of the six consortia systems. On this front, the alliance is once again uniquely qualified. The alliance has assembled a data conversion team that includes vendor expertise for the six California safe-haven systems. This brings unparalleled consortia system knowledge to the CCSAS CSE project for 100% of the statewide caseload, a claim no other vendor can make.

As a result of transitioning to one of California's safe haven systems, many counties have already worked with us through this process and have used the automated toolset. This familiarity will translate into a quicker conversion startup and a lower overall conversion risk.

The alliance is keenly aware that minimizing LCSA disruption, both before and after cutover to the statewide system, is a key goal. To that end, our proposed conversion strategy seeks to combine tried and true methodologies and toolsets together with deep consortia system expertise to reduce dependence upon, and disruption to staff and operations.

3C.13.1.2 Approach

3C.13.1.2.1 Conversion Strategy - Phase I vs Phase II

As described earlier in our approach, our overarching strategy calls for a two-phased implementation approach. The Statewide Services component, Version 1, of our solution is built and deployed during Phase I followed by Phase II, statewide rollout of the Version 2 solution. Our proposed conversion strategy is framed by this two-phase project approach.

The Phase I data conversion strategy transitions all non-CASES, non-ARS LCSAs to the CASES consortium. Specifically, data from the 14 LCSAs that comprise the KIDZ and STAR/KIDS consortia will be converted to CASES-specific formats. At the end of Phase I, 55 LCSAs will operate the CASES system while three LCSAs, Los Angeles, Orange, and San Diego will continue running ARS. Summary case data from the CASES and ARS databases will be extracted and bridged to an integrated database structure within the Statewide Services component. In addition, non-IV-D orders from the Non-IV-D Information Collection and Evaluation (NICE) database will be incorporated to provide

the State with a single repository of converted IV-D and non-IV-D cases and orders (updated per CR-00124a).

Phase II of the CCSAS CSE project focuses on the deployment of a single statewide system built around best of breed technologies and child support program management practices. The Phase II data conversion strategy is aimed at converting data from the local, LCSA-specific ARS and CASES databases to the new single statewide repository along with data from other State-level repositories (e.g., the California Central Registry and the Paternity Opportunity Program (POP) Database) (updated per CR-00124a).

The proposed conversion methodology and automated toolset will be similar for both phases of the conversion effort. Accordingly, the remainder of this section is organized as follows:

Common conversion methodology
Historical data archival and retrieval
Phase I specifics
Phase II specifics
Managing conversion
Conversion risks and mitigation strategies

As we've moved through California, we've refined our ten-step process to reduce risk and gain efficiencies, especially during the iterative automated data cleansing activities.

3C.13.1.2.2 Common Conversion Methodology

The alliance's proposed common conversion methodology (for Phase I and Phase II) has been developed and refined through our successful execution of over 35 LCSA transitions. As we've moved through California, we've refined our ten-step data conversion process to reduce risk and gain efficiencies, especially during the iterative automated data cleansing activities. Our toolset is robust, yet simple to use and well understood by staff that have been involved in one of the state's safe-haven transitions. The ten steps of our proposed Conversion Methodology and Toolset (CMAT) are:

1. Prepare Data Conversion Plans.
2. Establish conversion requirements
3. Prepare data maps
4. Conduct gap analysis
5. Prepare detailed LCSA-specific plans
6. Develop data derivation programs
7. Iterative data submission and automated data cleansing
8. Data acceptance testing
9. Cutover
10. Post-cutover

Figure 3C.13.1-1 presents the conversion methodology process flow.

alliance Conversion Methodology and Toolkit
10-Step Conversion Process Flow

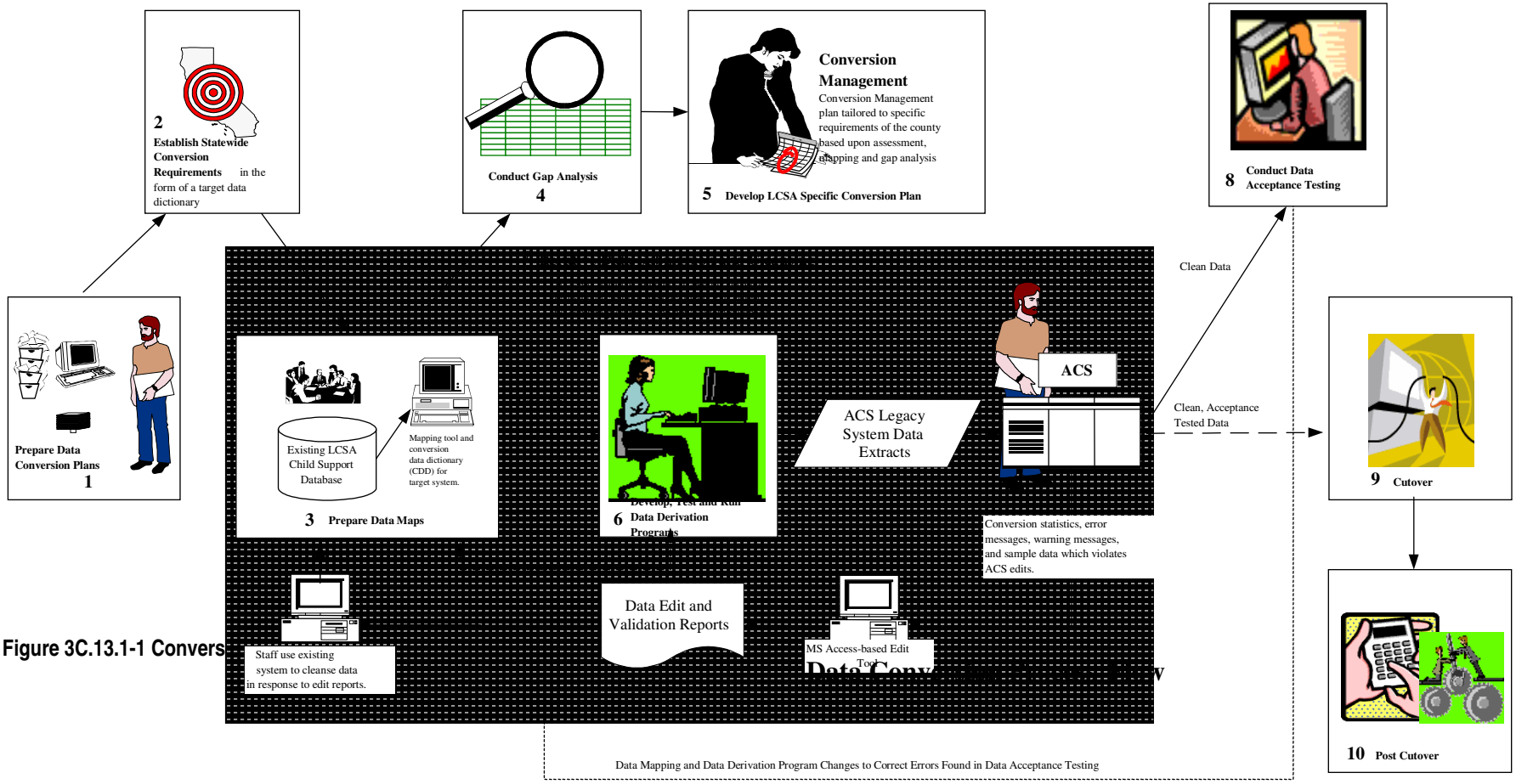


Figure 3C.13.1-1 Convers

1. Prepare Data Conversion Plans

The first step of the process is to prepare a data conversion plan (CDL TM 042) that will define the approach, methods, tools, and strategies for performing the actual conversion of data from legacy systems to the target application. In addition, roles and responsibilities will be defined for each organizational entity involved in the conversion process.

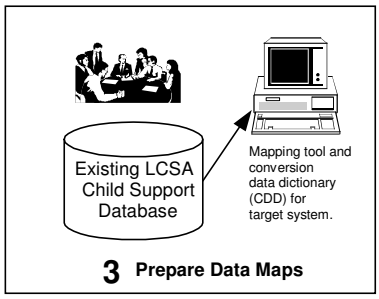
2. Establish Conversion Requirements

Next, conversion requirements are defined (CDL TM 041) in the form of the target data dictionary. This data dictionary includes a data dictionary hierarchy, descriptions of tables, table elements, field formats, edits and validations, and valid value codes for the new system. The target data dictionary, once established, is used as the target against which legacy data is mapped (i.e., Step 3). Legacy data within this conversion scope does not include historical data accessible for query only through legacy applications that have been retired.

Maintaining the integrity of the data dictionary over time will be of critical importance to the success of the conversion effort. As the development activities of the project progress, there will be changes and enhancements made to the new system database design. The alliance Data Conversion Team will participate in project-wide database design change control meetings and processes so that the target data dictionary used for data mapping remains in sync with the Development Team's database design effort. This project-wide process will be used so that changes made to the data model during the development lifecycle are evaluated for impact to conversion. This process meets the SCP requirements for aligning conversion team activities with the activities of the development team. As changes are approved and implemented, our Automated Data Mapping Tool and maps will be updated to comply with the updated target data dictionary.

Changes made to the database design that impact completed statewide data conversion requirements will be addressed through the change request process.

3. Prepare Data Maps

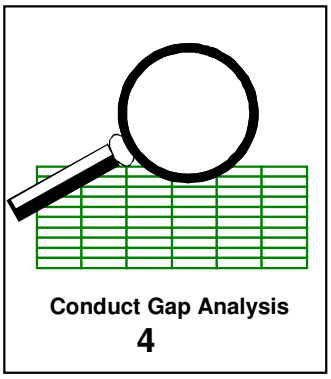


This step of the conversion methodology involves mapping the source LCSA data to the target data dictionary using the Automated Data Dictionary and Data Mapping Tool (ADD/DMT). The resulting data maps form the basis for subsequent design and development of derivation programs, which format data for processing through an ACS. It is during the mapping activity when the conversion strategy by data source is developed, culminating with the development of a single set of detailed data maps for each consortia system.

To begin the mapping activity, staff compares the data available from the LCSA legacy system to the data required by the target data dictionary. This comparison is done to identify any data structure or mapping related issues that must be resolved prior to

mapping. Using the ADD/DMT, our team then maps the legacy data to the target data dictionary, completing table-to-table mapping followed by element-to-element mapping. The ADD/DMT provides several reports that facilitate the identification and resolution of data mapping issues as well as support for the subsequent development of derivation programs.

4. Conduct Gap Analysis

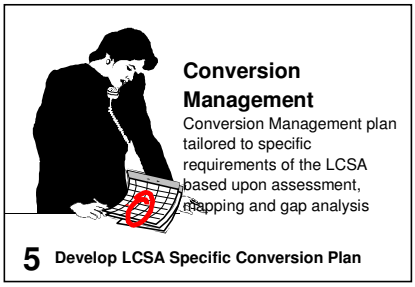


Once the data is mapped, a gap analysis is conducted to determine the extent to which required target data cannot be converted from automated sources. In those cases where a gap exists, our team will work with State staff to determine the most appropriate method of conversion.

For example, required target data might be maintained in manual form. In those cases where the volume of data justifies the effort, our team can develop simple data input screens and related capture tools that will ease the transformation of manual data into electronic format. By transforming the data from manual to electronic form, our team can clean and eventually convert the data using the ACS process, a much more efficient method than manual cleanup and conversion.

In those cases where required legacy data does not exist or is not accessible, our team will work with the State to define business rules for deriving the field values required for conversion. For example, the field values required in the target data dictionary may be derived using algorithms or by populating the target database with standard default values established in conjunction with the State.

5. Prepare LCSA-Specific Conversion Plans



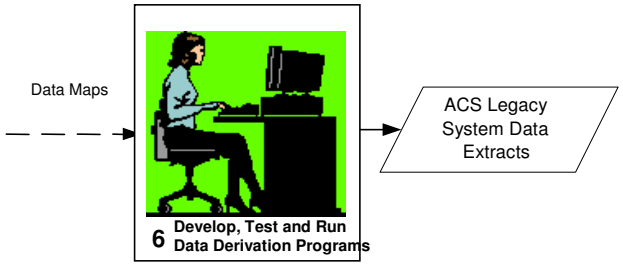
By step 5, our team will have developed a solid foundation of LCSA-specific information from which to develop a detailed LCSA conversion plan. The scope of the plan covers activities from the preparation of derivation programs through post-cutover.

The planning process involves first identifying an appropriate conversion method for the LCSA. Several factors are considered, including the availability of automated versus manual data, the validity of data in existing systems, and the availability of State staff for analysis of existing data. Second, processes are defined for extraction of data, identification and development of rules for handling data exceptions, and for loading the data into the Automated Conversion System (ACS). The ACS is used to convert legacy data to the target format.

Finally, planning for the final production data conversion requires an understanding of the volume of data being converted for each LCSA. Typically, conversion activities are planned over the weekend prior to system implementation. This provides enough time to extract and convert data. For the very large counties, the cutover timeframe may require that activities be scheduled over a 3 or 4-

day weekend plus two business days. Our team will work with State staff to confirm hardware, software, and storage resources are available to provide the required system performance and processing speed needed to complete conversion activities on time.

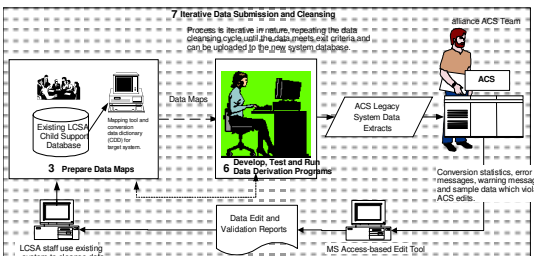
6. Develop Data Derivation Programs



Using the data maps as a primary input, our team will design and develop derivation programs to be used for the generation of legacy system data extract files. The State is responsible for providing data extracts for each LCSA. This is done to avoid issues associated with the Data Conversion Team working with LCSA production data. The extract formats specified within the mapping tool will be similar to the underlying table structures in the target data dictionary, resulting in the creation of target-like data extract formats.

Derivation programs will be thoroughly unit tested prior to use in the iterative automated data cleansing effort. The derivation programs are then run against the data provided by the county in the form of the data extract file. Once the automated data cleansing effort begins, the derivation programs continue to undergo modification as a result of data cleansing issues not originally considered in the data mapping process. In those cases, data maps are updated and derivation programs modified to accommodate new requirements. Each time the derivation programs are modified, unit testing is conducted prior to generating the data extracts required for subsequent iterations of automated data cleansing.

7. Iterative Data Submission and Data Cleansing



The successful conversion of child support data is largely dependent upon the quality of the data being converted to the new system. The goal of this step in the conversion methodology is to increase the percentage of cases that can be converted using automated tools and reduce the number of cases that must be converted manually at implementation. By using the ACS process described below, the bulk of the cleansing effort should be automated, significantly reducing resource dependencies upon State staff for data cleanup and manual entry.

The iterative data cleansing process involves the following steps:

Execute data derivation program to format LCSA legacy data for the ACS
Run the Automated Conversion System (ACS) process to identify rejected data
Use the Automated Data Cleansing Tool (ADCT) to report data rejects to the LCSAs

alliance and State staff work together to map a corrective action for the rejected data, which may call for correcting/modifying data maps, correcting/modifying derivation programs, or simply cleansing data in the legacy application
When remediation is completed, a new extract file is sent to the ACS team and the process is repeated

Our proposed conversion approach emphasizes converting data in discrete phases rather than all at once in order to increase the team's ability to quickly identify the root causes of and resolve data mapping or derivation program issues. Data from one phase (e.g., case and member information) is mapped, extracted, and cleansed prior to moving on to the next data phase (e.g., support order and financial data). Doing so reduces the possibility that data from earlier phases mask or complicate the resolution of data issues emerging in later phases.

To facilitate this building block approach, the conversion teams assigned for each LCSA generally produce extracts in the following order:

Case and member data;
Employer and other secondary table data;
Welfare data
Support order data
Detailed financial data

ACS applies major edits and validations required to convert data to the target database. Duplicate intra-LCSA cases or those that violate case constructs are rejected. Other major data problems may trigger the rejection of cases, but most errors encountered in an ACS run result in the rejection of a particular piece of the case data without rejecting the entire case. These are known as table rejects. A number of other edits and validations are applied to confirm the correct formatting of numeric data, dates, and valid coded values. Where possible, ACS attempts to assist by supplying data defaults for elements not provided in the LCSA data extract.

During Phase II, the conversion process will also involve the convergence of data from 58 different databases to a single common lookup table within the CCSAS CSE database. The alliance Data Conversion Team will use a "database of record" data validation table within the ACS process to validate the county data and generate reports indicating which case records require cleanup. For example, for employer data, the ACS process may use the EDD EIN database as the "database of record" against which county employer name and id data can be validated. Cases that include employer data that does not pass the EDD EIN validation step will be reported and referred to LCSA staff for correction prior to the next iteration of cleansing.

Each iteration of data processed through ACS generates a number of error and warning messages. ACS results are sent back to the LCSA via the Automated Data Cleansing Tool (ADCT). Using this tool avoids reliance on large paper reports, facilitates the rapid dissemination of ACS results to the LCSAs, and helps focus the attention of the local conversion support team on the most important errors encountered.

Wherever possible, the tool contains a sampling of the raw LCSA data that violates each of the edits or validations for which error and warning messages are produced. A State user or

programmer can generate examples of extract records that violated the edit/validation. Having examined the information, conversion staff can more quickly resolve the data or derivation program defect.

8. Data Acceptance Testing

8 Conduct Data Acceptance Testing

Step 8 involves the structured acceptance testing of data quality generated by the ACS process. Structured testing involves 4 general steps:

1. The alliance Data Conversion Team runs the ACS process and loads the converted data to the LCSA specific data acceptance testing environment database.
2. State staff executes data acceptance testing scripts to test the quality of the data converted through ACS. Test scripts are prepared by the alliance Data Conversion Team.
3. State staff review and validate the results of testing and refer defects to a joint team of alliance Data Conversion staff and State staff for tracking and resolution. This team performs a triage process that assigns each defect to one of three general categories: Data, Application, or Environment.
   - Data defects are further analyzed to determine whether the defect was generated by an incorrect mapping of data or by a defect in the data derivation programs used to prepare LCSA data for the ACS process.
   - Application defects are logged as a known defect and referred to the alliance Data Conversion Team. The alliance Data Conversion Team then coordinates with the application development team to log and track the resolution of the application defect. The application defect is identified as a known defect until the alliance Data Conversion Team receives notice from the alliance Application Development Team that the defect has been resolved.
   - Environment defects are used to identify and correct issues associated with the testing environment.
4. Once categorized, defects are assigned for correction. Resulting defect fixes are integrated into the testing environment for re-testing in subsequent cycles. Pre-Release notes are distributed prior to each cycle, detailing which defects have been resolved and which are still open. These notes provide testers with information that helps them focus their effort on re-testing fixes and identifying new defects.

This 4-step process is repeated until the test results meet the exit criteria defined for completion of data acceptance testing.

9. Cutover

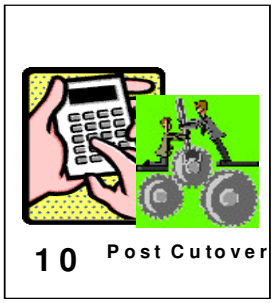
9 Cutover

Our approach assumes that we will be able to complete conversion runs within the window of a weekend for the vast majority of counties cutting over to the new system. For the very large counties, conversion might extend over a long weekend (i.e., Saturday, Sunday, and a holiday) plus 2 business

days. For those counties where business day downtime is expected, our implementation team will work with the State to define workaround processes that will enable operations to continue.

Our team has used simple operations continuance methods in previous conversions, including completion of screen print forms that look exactly like input screens of the new application. These forms are completed with updates to case data, then batched and key data entered by key data operations staff once the system cutover is complete. Our implementation team will consider using this and other methods in the development of a business operations continuance plan that fits the specific needs of the program.

10. Post-Cutover



Our focus at post cutover is to identify and resolve conversion fallout and data conversion errors. Facilitating this process, our CMAT Conversion Edit Tool generates conversion error reports that identify cases or partial cases that were not converted. Our team will use these reports to analyze any failed conversion attempts, document the planned approach for subsequent attempt(s) (such as additional data cleanup and rerun, or manual process), and execute and oversee reruns or manual processes as appropriate.

A post-conversion fallout report prepared after the production conversion is complete will provide information about what actually fell out of conversion. In addition, a county specific conversion reconciliation report will also be prepared and include:

Cases with member detail cross referenced from the old system to the new CSE system
Cases with member detail of balance information cross referenced from the old system to the new CSE system
Mapping of duplicate or merged data from the old system to the new CSE system
Pre-conversion and post-conversion trial balances

In preparing this report, the alliance Data Conversion Team will establish pre-conversion and post-conversion trial balances, and provide materials and support necessary for each county to conduct reconciliation. The alliance Data Conversion Team will conduct a summary total comparison check of balance and support terms in to the conversion process to summary total balance and support terms out of the conversion process. Out of sync conditions will be reconciled at the case level as required.

There may be instances where data will pass through the conversion process, but require remediation in the target system. The discovery of inaccurate data will occur within the normal course of post-conversion operations. As each data error is surfaced, our team will work with conversion support team staff to determine, on a case-by-case basis, a resolution strategy. In some cases, it will be appropriate to run Structured Query Language (SQL) scripts to repopulate an incorrectly populated field across cases. For isolated instances impacting one or a small number of cases, the resolution may be a simple online correction of data using the new application.

3C.13.1.2.2.1 Historical Data Archival and Retrieval

Our conversion strategy also addresses the issue of historical data retention. California's multiple interim system conversions to the safe-haven child support systems have created a substantial number of legacy systems. LCSAs continue to keep many of these systems running in query-only mode to access historical information. Primarily, this information is used to compute and defend disputed balances or to research financial activity on a case. While it is clear that a true business need exists to preserve this information, the State does not want to continue operating systems these systems for the long term. To address this issue, the alliance is proposing the following general approach:

Design, Construct and Implement a Legacy Data Archive

The alliance will design, construct and implement a new database (Legacy Data Archive) in accordance with the Legacy Database Archive Design Description (TM 100) to store targeted historical data currently maintained in the State's legacy systems. Once populated, users in the LCSAs will be able to access this database to view data converted from their old legacy systems. The intent is to establish a single, statewide repository to store historical data and thereby replace the numerous legacy systems currently operating throughout the state.

Critical to the success of this effort is the need to limit the amount and type of information to be stored in the Legacy Data Archive. As with all data conversion efforts, the value of preserving information must be balanced against the cost (most significantly human resources) of gettting data ready to convert. While it is likely that legacy systems vary widely in the breadth, depth, and volume of historical information they store, only a subset of the data must be preserved. Accordingly, data to be stored in the Legacy Data Archive will focus on key case, member, and financial history data[70] (i.e. information needed to reconstruct a balance) and will include the following types of information:

- Case History;
- Member History;
- Welfare History;
- Receipt History;
- Disbursement History;
- Adjustment History;
- Interest Terms History;
- Intercept History;
- Recored Order History;
- Legal History; and
- Significant Events.

The specific data elements/content of the Legacy Data Archive will be determined jointly with the State and documented in the Legacy Database Design Description (TM100). Thereafter, specific legacy data archive requirements will be documented in the Data Conversion Requirements Document (TM041). Business rules for the Legacy Data Archive will also be defined via the Legacy Data Archive Plan (TM040) to specifiy which population of historical cases will be candidates for conversion.

Build a Legacy Data Archive Automated Conversion System and Supporting Tools

Unlike data converted for production use, an emphasis will not be placed on strict data editing and data cleansing for information bound for the Legacy Data Archive. The goal is to make historical

[70] System-generated financial transactions will not be converted as they can be derived during the balance regeneration process (e.g. arrears accruals, distribution transactions, etc.)

data easier to access and manipulate, but not to perform significant data cleansing at the time of conversion. To do so would potentially create a substantial data cleansing backlog for State staff for data that is not required for day-to-day operations and may not be readily modifiable[71]. Therefore, historical data will generally be converted to the Legacy Data Archive in its current state of cleanliness and later audited via the best of breed system as business events warrant.

Even with this premise, there is a need to perform some editing of historical data to confirm that it meets the data conversion requirements defined for the Legacy Data Archive. Therefore, an Automated Conversion System (ACS) will be developed to perform edits against data bound for the Legacy Data Archive. Data that fails to meet edits performed by the ACS will be loaded to the Automated Data Cleansing Tool (ADCT) and reported back to the LCSA of origin for further analysis. The design and build of the Legacy Data Archive automated conversion tools will be documented in the Legacy Data Archive Plan (TM040).

Map and Derive Historical Data From Legacy Systems

The alliance will develop standard format descriptions that will describe how data bound for the Legacy Data Archive must be submitted. Using this baseline standard, the alliance will map and derive historical data stored in the KIDZ and STAR/KIDS consortia systems in Phase I, CASES and ARS in Phase II. Data maps will be developed using the Automated Data Dictionary and Data Mapping Tool (ADD/DMT) while new programs will be written and tested to derive data into the standardized formats. Once prepared, the data maps and derivation programs will be re-used for all counties within a consortium.

The alliance will also provide support to assist the State with converting historical information on systems outside the alliance's responsibility. The alliance will provide one full-time support staff to assist the State by performing tasks such as responding to data mapping issues and interpreting the results of errors generated by the ACS. The alliance will provide historical data conversion support beginning in month six of the project and continue through month sixty.

Cutover to the Legacy Data Archive

During Phase II, historical data will be converted to the Legacy Data Archive at or near the same time the LCSA is cutover to the new best of breed system[72]. The conversion of historical data to the Legacy Data Archive will not hold up an LCSA's scheduled cutover to the Version 2 CSE system. In other words, historical information available at the time of an LCSA's scheduled cutover to Version 2 will be loaded to the Legacy Data Archive. Thereafter, the State may begin the process of retiring the use of the legacy system within the LCSA.

This same approach will be followed for LCSAs converting to CASES during Phase I. However, if an LCSA is scheduled to convert to CASES prior to the implementation of the Legacy Data Archive (including query access), the alliance will come back to the LCSA to convert historical information. Doing so will require that all such LCSAs continue to operate their legacy system in query mode until historical information is converted to the Legacy Data Archive.

[71] Legacy systems typically provide query-only rather than update access.
[72] For counties with large data volumes, the cutover to the Legacy Data Archive may be scheduled to occur just after the cutover to the best of breed system to avoid creating unnecessary downtime in the county.

3C.13.1.2.3 Phase I Specifics

As discussed in Section 3C.13.1.2, Phase I of our overall solution calls for the 58 counties to operate on either the CASES or ARS systems linked to a new Version 1 Statewide Services component. The 14 counties that comprise the, KIDZ and STAR/KIDS consortia will be transitioned to the CASES consortium. The scope of this effort requires converting approximately 24% of California's counties and 29% of the statewide caseload. The conversion approach for Phase I does not have any impact on the existing CASES databases.

Consortia	Total Counties	Total Open Caseload[73]
KIDZ	10	214,154
STAR/KIDS	4	369,185
Totals	14	583,339

Figure 3C.13.1-2 Counties and Caseloads for the Phase I Implementation – *This table provides the specific number of counties and total caseload to be converted for each of the consortia.*

Our proposed Phase I conversion strategy also envisions converting non IV-D order information from the NICE system database. Since October of 1998, DCSS and the Clerks of Court in local jurisdictions have worked together to capture information on non IV-D orders. Current procedures require the Clerks of Court to send a manual form to DCSS each time a new non IV-D order is established. DCSS then key enters data from the form, which is then stored in an automated format. Our plans call for converting the NICE database into the Statewide Services repository via an automated interface. Non-IV-D orders will be edited prior to being loaded. Rejected data will be returned to DCSS for manual data cleansing and re-submission (updated per CR-2-00124a).

The alliance will utilize the conversion methodology described earlier in this section to convert IV-D data from the 14 transitioning counties. However, the alliance intends to re-use several key existing conversion components used successfully during our prior CASES conversion experiences. By leveraging the following existing assets, our Phase I conversion approach significantly reduces the overall conversion scope, timeline, and most importantly, risk.

CASES Automated Data Dictionary and Data Mapping Tool (ADD/DMT): This tool currently documents the specific data conversion requirements for transitioning a LCSA to CASES. Put simply, the conversion target has already been described and therefore data mapping tasks can begin early.
CASES Automated Conversion System (ACS): The alliance assumes that the state will reuse the ACS process for editing and validating data bound for the CASES target database. The

[73] Open caseload counts from the Sept 2000 Master Case List.

ACS software includes hundreds of data editing rules to validate that data passed into the target CASES environment meets the CASES conversion requirements.

The re-use of these two important conversion tools means that our conversion team will have a sizeable head start on the 14 LCSA transitions. In addition, project risk will be reduced significantly through the re-use of software and tools that have time and again shown to be effective in converting counties to CASES.

To introduce further efficiencies into the conversion process, our conversion teams will prepare one set of data maps and data derivation programs for each consortium. These data maps and derivation programs will be developed by working with the lead LCSA for each consortium. Thereafter, the resulting maps and programs will be re-used for the remaining counties within a consortium. Working in this fashion will produce consistent results across counties while limiting the time required of State staff.
A dedicated alliance Data Conversion Team will be assigned to convert data from the NICE database into the Statewide Services repository. The conversion of non IV-D will be similar to the methodology described earlier except that there will be no LCSA involvement nor will a gap analysis be required. Rather, the alliance will work with State staff to complete the conversion effort.

Figure 3C.13.1-3 [RESERVE]

3C.13.1.2.4 Phase II Specifics

Phase II of the CCSAS CSE project focuses on the deployment of a single statewide system built around best of breed technologies and child support program management practices. The Phase II data conversion strategy is aimed at converting data from the ARS and CASES databases to the new Statewide Services repository. Additionally, data will also be converted from State-level sources including the California Central Registry (CCR), POP and IDB databases (updated per CR-00124a).

The Phase II conversion approach will adhere to the methodology described earlier in the Approach section. However, as with Phase I, our conversion team will have a head start on the statewide transition. This advantage stems from our plans to leverage the Federal Case Registry (FCR) interface between the ARS and CASES consortia and the Statewide Services component of our Phase I solution.

The ARS and CASES consortia (as well as other consortia) submit data to the State via a Federal Case Registry (FCR) extract. This information is then passed through the FCR ACS to edit data prior to its submission to the Federal Parent Locator Service (FPLS). Data rejects identified by the FCR ACS are communicated back to the counties via our Automated Data Cleansing Tool (ADCT).

The similarity between the FCR process and our proposed conversion methodology provides an opportunity to leverage the FCR infrastructure for use during Phase II conversion. Specifically, our Phase II plans for conversion include:

Leveraging the consortia FCR extracts: The current FCR extracts are primarily case and member information – the first logical step in converting a case. Therefore, rather than starting

from scratch, we plan to build on this data foundation by adding new conversion requirements for information like financial, welfare, and employer data. This building-block approach of adding new data groupings to the FCR extract will ultimately result in a full caseload, full data extract for use in converting to the best-of-breed system.
Leveraging the FCR ACS: The current FCR ACS performs numerous data edits and validations including checks for valid case constructs, duplicate cases, and duplicate members. Again, rather than building a new ACS for the best-of-breed solution from scratch, the alliance plans to expand the FCR ACS in parallel with the FCR extracts. For example, as financial data is added to the FCR extract, the FCR ACS will be enhanced to include edits that validate financial data. This cycle will continue until the data required for the best-of-breed solution is fully converted. Our plans, however, do call for migrating the FCR ACS off the current VAX platform to a high-performance, Intel-based environment. Doing so will vastly improve CPU performance and critical disk I/O speed for converting large counties.

As shown in figure 3C.13-1-4, the re-use of these key existing assets will provide our conversion team with a solid base to build from. In addition, LCSAs are likely to be highly supportive of a conversion approach that takes advantage of work already performed by DCSS for the FCR interface.

Figure 3C.13.1-4 [RESERVE]

Image Conversion

The alliance imaging solution is described in detail in Section “4A.3.2a: CSE System Architecture - Enterprise Viewpoint”. We understand that there are LCSAs today that use imaging as part of their current CSE implementation. As discussed Section 4A.3.2a, if elected by the LCSA, current imaged documents of open cases may be converted (transferred) to the CCSAS CSE imaging platform. The plan for an imaging conversion is described below:

JAD sessions conducted by the alliance will result in the definition of the image conversion process, requirements, timeframes, assumptions and roles and responsibilities for each LCSA. The alliance has assumed that there are 10 LCSAs that currently use image technology for their open cases.
The State is responsible for providing images to the alliance in an agreed to industry standard format (TIFF, PDF, IOCA, MODCA, JPEG or GIF) along with the indices that accompany the image file (i.e. case number, form type, timestamp, etc).
With the scope established, the alliance sizes the effort to perform the conversion of images and indices and import them into the CCSAS CSE application. The size of this effort will not be known until the JAD sessions have been conducted and requirements collected.
The alliance will be responsible for batch loading the images and indices into the CCSAS CSE system.
Once converted (transferred) the images will be accessed in the same manner as new incoming images with the proposed CSE application and a Web browser.

3C.13.1.2.5 Managing Data Conversion

3C.13.1.2.5.1 Managing Conversion Software and Hardware Configuration

Conversion Software Configuration Management

The conversion software development environment will be managed using standard methods of library management and version control. This section describes the software configuration management and coding standards to be used on the CCSAS CSE project.

Code Library and Version Control

ACS developers are required to use the conversion code library and version control systems to safeguard code, and document code changes and variants throughout the life of the conversion project. Use of a code library with version control provides the ability to “back out” changes as required to retain configuration integrity.

Example Directory Structure

The following provides an example of the directory structure developers will be required to follow:

ACS directories:	
CCSAS_LIBRARY	Nightly batch script files. Used to automate migration between development, production, etc.
CCSAS _SRC	Code repository for current version of source code.
CCSAS _COPY	File layout include files.
CCSAS _DATA	Static data files used within ACS.
CCSAS _EXE	Executable files.
CCSAS _LOG	Log files for ACS management processes (movement of code to from development to production etc.)
CCSAS _OBJ	Object modules.
CCSAS _PROD	Operating system scripts used to run ACS.
CCSAS _UTIL	Utility directory contains non-production scripts, helpful utilities.

Figure 3C.13.1-5 ACS Directories – *Consistent ACS Directory Structure Development Reduces Functionality Risk.*

County Specific Directories:	
CTY_ROOT	Root directory for county. Contains county specific configuration and environment variables file. Static directory, files remain between iterations.
CTY_CTL	Directory for cross-reference files.
CTY_RAW	Contains input data to be processed by ACS.
CTY_EDIT_EXE	Contains executables for error processing.
CTY_EDIT_IDX	Directory for error index data.
CTY_EDIT_OBJ	Contains object modules for error programs.
CTY_EDIT_OUTPUT	Output directory for conversion statistics and edittool data files.
CTY_EDIT_SRC	Contains derived programs if needed.
CTY_IDX	Output directory for index data.
CTY_LOG	Contains log files for processes run during an ACS iteration.
CTY_OUTPUT	Output directory for C-IV upload data.
CTY_SSEQREC	Sorted sequential records directory.

Figure 3C.13.1-6 County-Specific Directories – *Consistent County-Specific Directory Structures Provides Low-Risk Data Transfer to the State CSE System.*

Shell Programs

Conversion developers are provided standard "Shell" programs on which to base new development. The standard shell programs are developed at the start of conversion construction and will be leveraged throughout the remaining portions of the ACS construction process. The basic types of shell programs are anticipated to be:

Load Routine – Standard ACS program. Reads in one or more sorted sequential files, performs edits and writes output.
Index Creation Routine – Reads one or more raw data files, writes subset of data to index output file. Loads data to database via mass update, performs add index function.
Reference Edit Routine – Reads and creates intermediate output data for use by other ACS routines.
SseqRec Creation – Merges one or more files into one sorted sequential record file (SseqRec) with record type codes. Most processing in ACS is performed using SseqRec data.

The use of standard shell programs will result in consistent and uniform code throughout ACS.

Example Generated File Structures

The file layouts for the majority of ACS processing will be derived using the data dictionary tool. These structures include both input and output file layouts. The use of automatically generated file structures will result in consistency of table and element names not only in the automated conversion tools, but also in the ACS code modules. Changes to the CCSAS CSE data dictionary are automatically reflected in the generated file structures. Often, when the changes are simple element size changes the conversion code can be updated simply by rebuilding the application with the generated file structures.

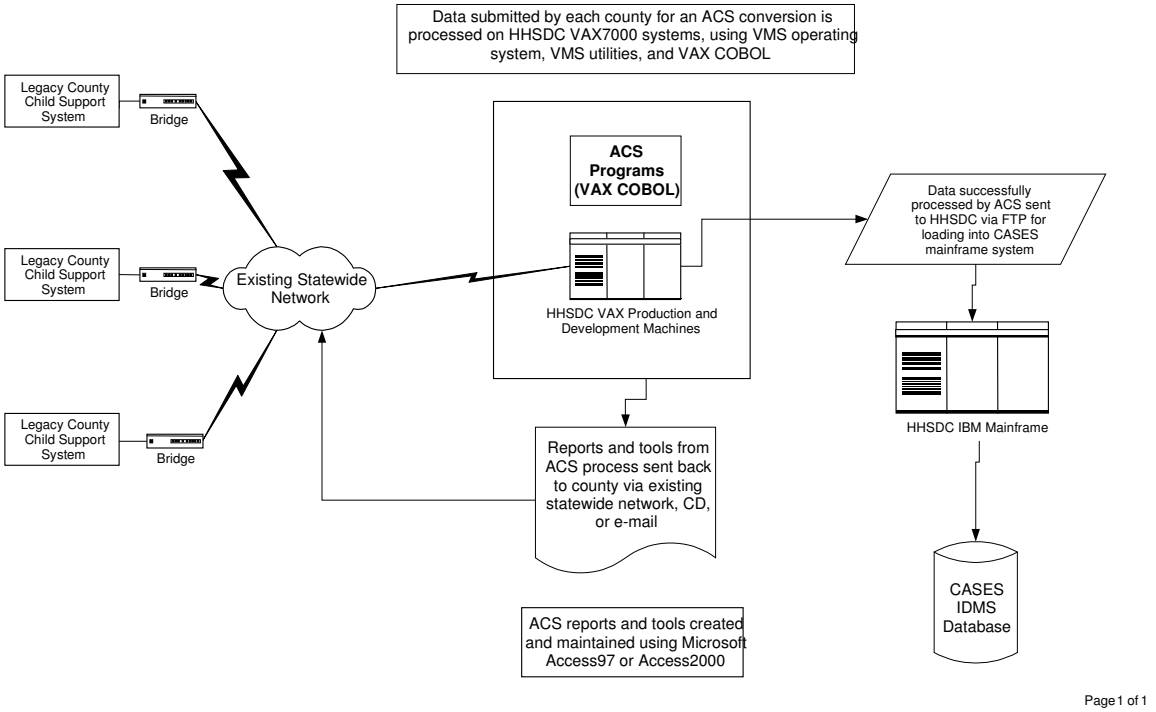
Code Values

Coded value reference tables will be created for most coded value elements and be maintained outside of the conversion code routines. It is likely that the coded reference tables for use in conversion routines will be maintained in an automated fashion through the data dictionary tool and the reference files will be automatically generated.

Conversion Hardware Environment

The hardware environment supporting the conversion management system is presented in Figure 3C.13.1-7 for Phase I and Figure 3C.13.1-8 for Phase II. Maintenance of the Phase I hardware environment remains with HHSDC. For Phase II, the conversion hardware environment will migrate to a Win-tel platform and be maintained by the alliance.

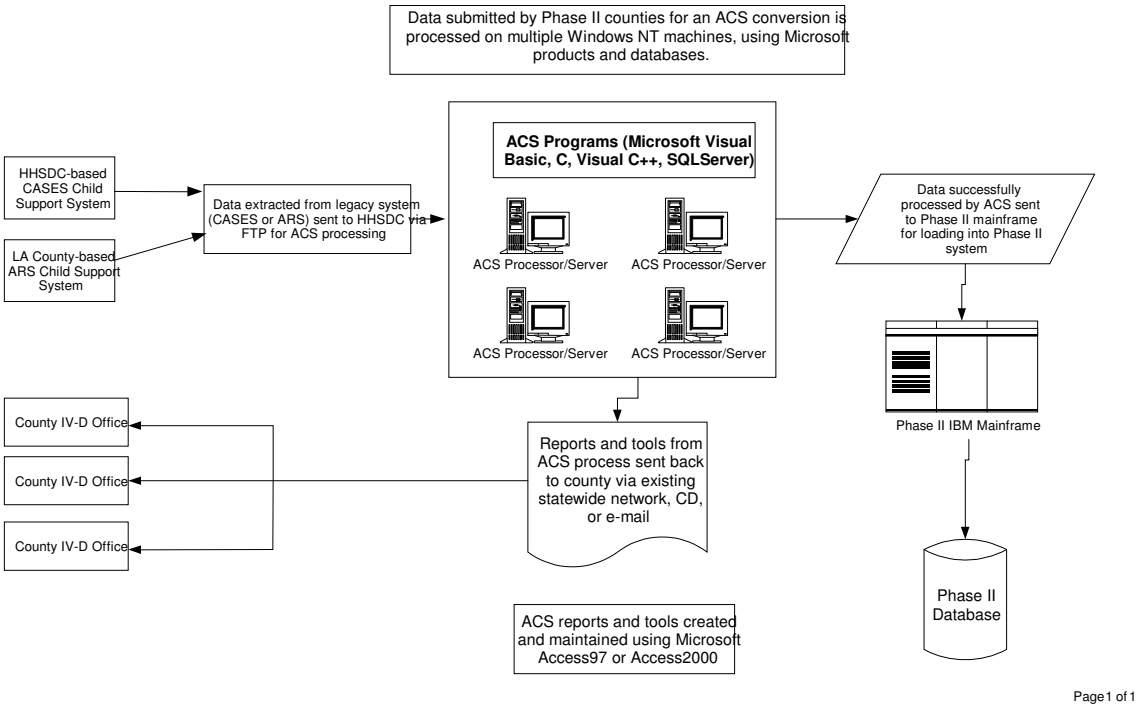
Phase I Conversion Hardware Environment



Last change:12/21/0110:05:00 PM

Figure 3C.13.1-7 Phase I Conversion Hardware Environment - *The hardware environment supporting the conversion management system.*

Phase II Conversion Hardware



- Last change12/21/0110:37:02 PM

Figure 3C.13.1-8 Phase II Conversion Hardware Environment - *The hardware environment supporting the conversion management system.*

Asset Management

As described in Section, 3C.13.4, the alliance approach to hardware configuration management is based upon the IBM Asset Management (AM) methodology. A complete discussion is provided in Section 3C.13.4.

3C.13.1.2.5.2 Data Conversion Roles and Responsibilities

Figure 3C.13.1-9 [RESERVE]

Figure 3C.13.1-10 summarizes the team functions:

The Data Conversion Functions
• alliance Data Conversion Management - The alliance Data Conversion Team will track and resolve key conversion issues, set project priorities, monitor progress, conduct status meetings, identify action items, and prepare periodic progress reports to the CCSAS CSE project management team. As the conversion activities progress LCSA by LCSA, the plan will be updated and refreshed to reflect current project requirements as well as lessons learned to date.
• alliance LCSA Data Conversion Support Team – The alliance LCSA Data Conversion Support Team leads the data mapping activities and develops derivation programs required to produce LCSA data extract files. In addition, this team provides defect tracking and resolution support during data acceptance testing. The alliance conversion support function will be structured into sub-teams that are consortia-governed. This approach enables each team to be focused on understanding the data structures of just one consortia system and be able to achieve higher data mapping efficiencies in subsequent LCSA data mapping efforts conducted within the same consortium.
• alliance Centralized Data Conversion Team – The alliance Centralized Data Conversion Team is composed of the alliance ACS Team, the alliance Target Data Dictionary Team and the alliance Data Acceptance Testing Home Team. • alliance ACS Team: Responsible for running the ACS process, developing the automated Data Dictionary/Data Mapping Tool, executing the Automated Error Reporting process, developing Data Capture Tools and Manual Conversion Tools as needed, and generating Financial Reconciliation Reports • alliance Target Data Dictionary Team: Responsible for maintaining the target data dictionary. • alliance Data Acceptance Testing Home Team: Responsible for data acceptance test script creation and maintenance, ACS run validation activities, data acceptance testing support to the alliance Conversion LCSA Support Team, and application defect tracking and coordination with the CCSAS CSE application development team
• State staff - State staff are responsible for participating with the alliance Data Conversion Team in data mapping and cleansing activities. State staff are responsible for data acceptance testing script execution and test results validation activities. In addition, State staff will be required to support cutover activities. Conversion as an implementation discipline requires a commitment of resources from each LCSA in order to successfully execute conversion activities. Our approach assumes that State staff estimates included in this contract represent a consistently staffed team of State staff dedicated to data mapping, data cleansing, acceptance testing and cutover support.

Figure 3C.13.1-10 The Data Conversion Functional Organization – *Clear understanding of the roles of the Data Conversion team facilitates communication between team members.*

Data Conversion Process Flow

3C.13.1.2.5.3 [RESERVE]

3C.13.1.3 Conclusion

Our proposed data conversion solution focuses on minimizing implementation risk and maximizing efficiencies in the overall process (from data mapping to the final load of converted data) via the re-use of proven methodologies and tools. The alliance and DCSS have already successfully partnered to deliver over 35 California counties to their selected safe-haven system. We are confident we can repeat this success as California moves to a single, statewide system.

3C.13.2 Transition Management

3C.13.2.1 Overview of Transition Management

Section 3C.13.2, Transition Management, describes the alliance approach, methods, and tools to implement the new CCSAS CSE systems. Our approach is structured into two phases, mirroring the overall project approach, and has been designed to facilitate the transition of users and functions in a way that promotes process uniformity and limits program disruption.

- **Phase I Transition** is focused on implementing Version 1 of our solution, which includes the State Case Registry, locate processing, federal reporting, and an interface to the Statewide Disbursement Unit (SDU). At the local level, 14 LCSAs will be transitioned from the KIDS, and STAR/KIDS consortia to the CASES consortium, creating a two-system configuration (CASES and ARS) for California.
- **Phase II Transition** is focused on rolling out Version 2 of the statewide child support system.

This section concentrates on the four of the seven implementation disciplines and provides the strategy for transitioning users and functions. It is organized as follows:

- 3C.13.2.2 Change Management
This section describes the approach and strategy for promoting process uniformity and managing the impact of change on users and external entities. Derived through lessons learned from CWS/CMS, it describes the scope of change management and the issues considered in the development of our change management program.

- 3C.13.2.3 Implementation Management
This section provides an overview of our implementation management approach, including how we organize the overall implementation team and how the implementation teams work in the individual counties undergoing transition to the new systems. We also provide a definition of implementation roles and responsibilities.

- 3C.13.2.4 Transitioning Users and Functions
This section describes the strategy for transitioning users and functions and the sequence in which the benefits associated with Phase I and II may be achieved. It defines the proposed timing of each transition, including the issues considered and assumptions made. It also describes how the approach is designed to limit disruption to users and customers.

- 3C.13.2.5 On-Site Support

This section describes the approach to providing support to LCSA and DCSS during transition. This includes the scope of support provided and the period of time support is available.

- 3C.13.2.6 Local Interfaces

This section describes how local county interfaces are established. These include the welfare interface, auditor/controller, and court interface where applicable.

3C.13.2.2 Change Management

3C.13.2.2.1 Overview

The alliance change management strategy is tailored to address both the scope and complexity of the project as well as the statewide reach of the project's impact. Change management, as a discipline, is an integral component of transition management, helping to reduce the risks of implementing a large-scale system. The alliance's change management strategy has three main goals:

- **Facilitate System Functionality Mapping in Phase I** – In Phase I of the CCSAS CSE project, 14 counties transition from one of two existing consortia systems to CASES. This transition provides system functionality maps so that the State can determine the impact on each of the 14 LCSAs of the CASES transition to current business processes.
- **Promote Business Process Uniformity in Phase II** – With the statewide implementation of the CCSAS CSE Version 2 application, the state has an opportunity to standardize business processes across the program. The primary focus of the Phase II change management strategy is to define the target set of standard CCSAS CSE business processes and then use those standards to guide implementation activities statewide.
- **Promote Project Understanding Through Outreach and Orientation During All Phases** –The communications effort of both project phases is a fundamental component of the project. Communications efforts include statewide outreach, local outreach, and orientation for users. These communications are essential to sustaining project momentum and maintaining a healthy project image.

Change Management Objectives

Our proposed approach for the CCSAS CSE change management aims to meet the following objectives:

Change Management Objectives
Define and implement uniform and standardized business processes supporting the CCSAS CSE Version 2 solution, and communicate the associated benefits.
Involve staff in critical activities, including the analysis of As Is and To Be business processes.
Develop local CCSAS CSE expertise prior to each LCSA implementation.
Support CCSAS project, DCSS and LCSA leaders in demonstrating their commitment to the strategic importance of the CCSAS CSE project.
Educate users on the importance of the CCSAS CSE project and associated changes.
Provide appropriate and well-timed information necessary to external entities to gain their acceptance and buy in.

Figure 3C.13.2.2-1 Change Management Objectives -*The objectives of change management are accomplished through business process analysis and communications.*

3C.13.2.2.2 Change Management Strategy

Our Change Management strategy is organized into two general types of activity: Business Process Analysis and Communications. Communications is further organized into orientation, local outreach, and statewide outreach. This section provides both an overview of our approach as well as the details of Phase I and Phase II and is organized as follows:

- Business Process Analysis
- Communications
- Orientation
- Outreach
- Phase I Vs. Phase II
- Phase I Specifics
- Phase II Specifics

Business Process Analysis

Business Process Analysis activities define and implement processes that work together with the CSE application in use by the LCSA. In Phase I, the aim of this activity is to analyze existing LCSA business processes in 14 counties as required to support the transition to CASES. In Phase II, the aim is to achieve uniform and standardized processes across the Child Support program, and support implementation of the CCSAS CSE Version 2 application.

In both phases, the alliance begins by mapping the functionality offered by the legacy system to functionality offered by the target system. This information is used to analyze the current business processes and identify changes that must be made to accommodate differences between the current and target CSE applications. In Phase I, the analysis is provided to the State to make implementation decisions used to guide the LCSAs' transition to CASES. In Phase II, the analysis results in the development of "To-Be" processes for each LCSA to support the transition to the CSE Version 2 application and serve as the blueprint for the CCSAS Business Model.

Communications

Clear and effective project communications play a critical role in smoothing the process of collaboration in key project activities. To the extent that LCSA and external entities understand the project, the better prepared they are to contribute toward the project's success. Additionally, it is important to have a coordinated communications effort to allow for consistency of message, especially in a project such as CCSAS with diverse groups who are interested in the project dispersed throughout the state. Communications fall into two categories: 1) internal orientation activities and 2) external outreach activities. The following table describes our guidelines for the development of orientation and outreach plans.

Communications Guidelines
Focus on the goal – We craft messages to support an organizational culture in which people and groups demonstrate attitudes, behaviors, and values in line with the CCSAS Mission.
Employ existing network – We build on established communications networks to facilitate formal and informal communications.
Share information openly and honestly – We share accurate, available information about the changes taking place. We work to create an environment of openness and trust that leads to a shared commitment to continuous improvement.
Reinforce key messages – Our communications work supports and guides users through the stages of change. We tell people what they need to know to do their jobs and be effective in the new environment.
Deliver messages sensitive to time and audience – We deliver messages in a time-sensitive manner to prevent confusion and/or rumor spreading. We provide the right information to the right people at the right time throughout the implementation process.
Measure Results – We monitor and evaluate progress and make adjustments to our communications approach as needed.

Figure 3C.13.2.2-2 Communications Guidelines - *The development of orientation and outreach plans form the heart of our communication guidelines.*

Orientation

The CCSAS CSE project orientation process delivers change messages to internal audiences, specifically those users with access to the new CCSAS CSE systems. Orientation consists of communications and involvement activities targeted internally on current child support systems users, including the DCSS and LCSA offices. The aim of orientation is to involve, educate, and facilitate the ongoing exchange of information necessary to help obtain buy-in and acceptance from program staff needed for the successful transition to the new environments.

Outreach

Outreach to parties affected by the CCSAS CSE project is vital to promoting widespread understanding and support for the various project initiatives. The fundamental purpose of the Outreach Plan is to identify outreach entities, generate awareness, provide consistent information, and eventually guide external entities to adjust to and incorporate required changes to their impacted activities. Outreach activities include the following groups:

- California State Legislature
- Governor's Office
- Office of Child Support Enforcement
- Department of Justice
- County Boards of Supervisors
- County Courts
- Human Services Agencies (IV-A)
- Unions
- Employers
- Advocacy groups
- Customers (case members)

Leadership

DCSS and LCSAs look to the leadership of the CCSAS CSE project as a gauge of project importance in the larger scheme of competing priorities. The momentum of a long-term project like CCSAS can only be sustained through demonstrative and committed project leadership, or sponsorship. Project sponsorship is the public participation of leadership in outreach and orientation activities that:

- Introduce and promote the change effort
- Assist in the definition of goals and expectations
- Deliver key messages in a consistent manner
- Make key decisions
- Demonstrate commitment to the change effort

Our approach involves working closely with the CCSAS CSE leadership to help them identify and deliver important messages to external entities while also delivering key messages to internal program users. Given the size and diversity of California's Child Support Program, we have identified key leaders at a number of different levels in the program as follows:

California Child Support Program
• DCSS Director • DCSS Chief Deputy Director • DCSS Project Leader
CCSAS CSE Project
• Project Agent • CCSAS Project Directors
California Counties
• LCSA Directors

Figure 3C.13.2.2-3 CCSAS CSE Leadership - *Strong leadership will be critical to the project's success.*

Phase I versus Phase II

The alliance change management initiative is organized into two phases, in line with the overall structure of the CCSAS CSE project. Figure 3C.13.2.2-4 presents an overview of the alliance's change management approach for each phase.

Figure 1 - CCSAS Change Management Strategy Overview

Phase I - CASES Implementation		Phase II - Best of Breed CCSAS Application Implementation
CASES Version 1 CSE Application	***CSE Application***	**Version 2 CSE Application**
14	***Number of Counties Impacted***	**58**
Business Process • Support CASES transition in each LCSA **Communications** • Orient internal staff to changes associated with CASES implementation • Inform external stakeholders of changes to service processes	***Primary Objectives***	**Business Process** • Achieve uniformity of process across the state **Communications** • Orient internal staff to changes associated with CCSAS implementation • Inform external stakeholders of new service channels and changes to existing service processes
Business Process • Alliance provides LCSA with mapping of functionality from LCSA legacy system to CASES • LCSA conducts required business process transition planning and implementation activities **Communications** • Alliance provides templates for internal orientation and external outreach activities • LCSA customizes templates • LCSA delivers external outreach activities • LCSA delivers internal orientation sessions	***Change Management Strategy***	**Business Process** • Alliance develops generic As-Is process maps for CASES and ARS counties • Alliance facilitates development of the CCSAS To-Be business process maps with involvement from key DCSS and LCSA staff • Alliance builds CCSAS Business Model and corresponding model office based upon To-Be process maps • LCSA's conduct gap analysis between as-is processes and CCSAS To-Be business process maps with alliance support • LCSA's implement new CCSAS To-Be business processes with alliance support **Communications** • Alliance provides templates for internal orientation and external outreach activities • LCSA customizes templates • LCSA delivers external outreach activities • LCSA delivers internal orientation sessions

Figure 3C.13.2.2-4 Overview of the CCSAS CSE Change Management Strategy – *Two distinct phases of change management planned to provide business process analysis and communications activities.*

Phase I

In Phase I, change management activities support 14 counties in the transition to the CASES CSE application. To do this, the alliance develops two system functionality-maps, one for each of the two consortium systems to be replaced by CASES. These maps correlate the functions offered in each legacy system (STAR/KIDS and KIDZ) with the functions offered by the CASES application. The State uses the maps as the primary input to determining how the current business processes must change to work with the CASES application.

State personnel use the orientation materials to apply LCSA-specific mapping of old business functions and terminology to new business functions and terminology. State staff will use these maps to determine what decisions must be made to accommodate the LCSA transition to CASES. With input from the Orientation Guide (CDL TM 055) and support from the alliance, the State will perform the business process analysis and planning activities necessary to transition from KIDZ or STAR-KIDS to CASES.

In the area of communications, the alliance develops generic templates for both external outreach and internal orientation. These templates are used by State staff to meet particular local outreach requirements and internal orientation needs. The State will conduct the actual user orientation session, with participation from LCSA leaders, as they are the appropriate group to be facilitating the change. The alliance supports the State personnel in planning for and delivering the user orientation. The alliance also supports statewide outreach activities throughout Phase I.

Phase II

In Phase II, change management activities focus on achieving uniformity of process across the state and supporting implementation of the CCSAS CSE application in all 58 counties. Similar to Phase I, the alliance change management team will develop functionality maps that show how functionality in CASES and ARS is accomplished in the new CCSAS CSE application. Using these maps as a key input, the alliance facilitates the development of the CCSAS Business Model, which defines both the operational and business process flow standards to be used as a guide during CCSAS CSE implementation in each LCSA. This includes developing county-specific plans, customizing the templates, delivering communications, and implementing the changes.

The State uses the orientation materials to create county specific As-Is and To-Be business processes based on the CCSAS Business Model. The State personnel will customize the As-Is templates to reflect local processes, perform the gap analysis, assess impacts, and define an action plan to address the discrepancies, make assignments and execute action items. The alliance will participate in the effort by explaining the templates, providing instructions on how to perform the tasks, conducting working sessions, reviewing information and tracking progress, documenting the customized To-Be workflows, and supporting the effort through on-site coaching and facilitation.

Similar to Phase I, the alliance approach to communications leverages generic templates that can be used by State staff to meet their specific local outreach and orientation needs. Again, the State will conduct the actual user orientation session with LCSA leaders and support from the alliance to leverage the alliance's familiarity with the CCSAS CSE application processes. The alliance also supports statewide outreach activities throughout Phase II.

The proposed change management methodology is similar for both phases of the project with the exception that Phase I is more narrowly focused than Phase II. The remainder of this section is organized as follows:

- **Phase I Specifics**
- Business Process Analysis
- Orientation
- Outreach and Sponsorship
- **Phase II Specifics**
- Business Process Analysis
- Orientation

- Outreach and Sponsorship

3C.13.2.2.2.1 Phase I Specifics

The Change Management objectives of Phase I are to:

- Facilitate the transition of 14 counties to the CASES application
- Perform outreach with external entities to inform them of changes that will impact their operations or processes used to interact with the CSE program
- Orient LCSA and DCSS system users to changes associated with the Phase I initiative through outreach initiatives

Figure 3C.13.2.2-5 presents an overview of the Phase I Change Management Strategy.

Phase I Change Management Methodology

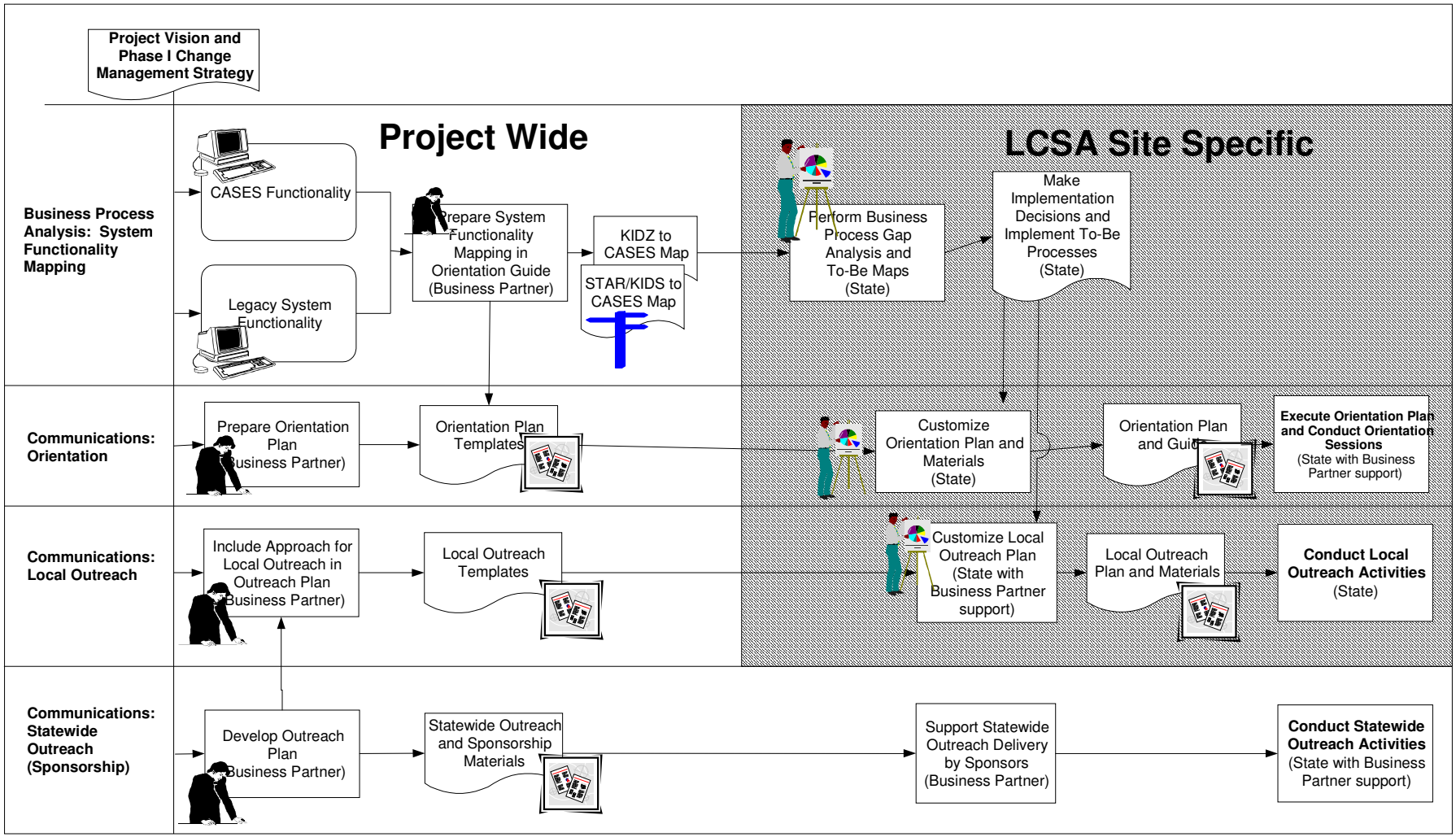


Figure 3C.13.2.2-5 Phase I Change Management Methodology – *Phase I change management provides early enhanced benefits by rapidly transitioning the LCSAs (with the exception of three) to the same system—CASES.*

Phase I Business Process Analysis: System Functionality Mapping

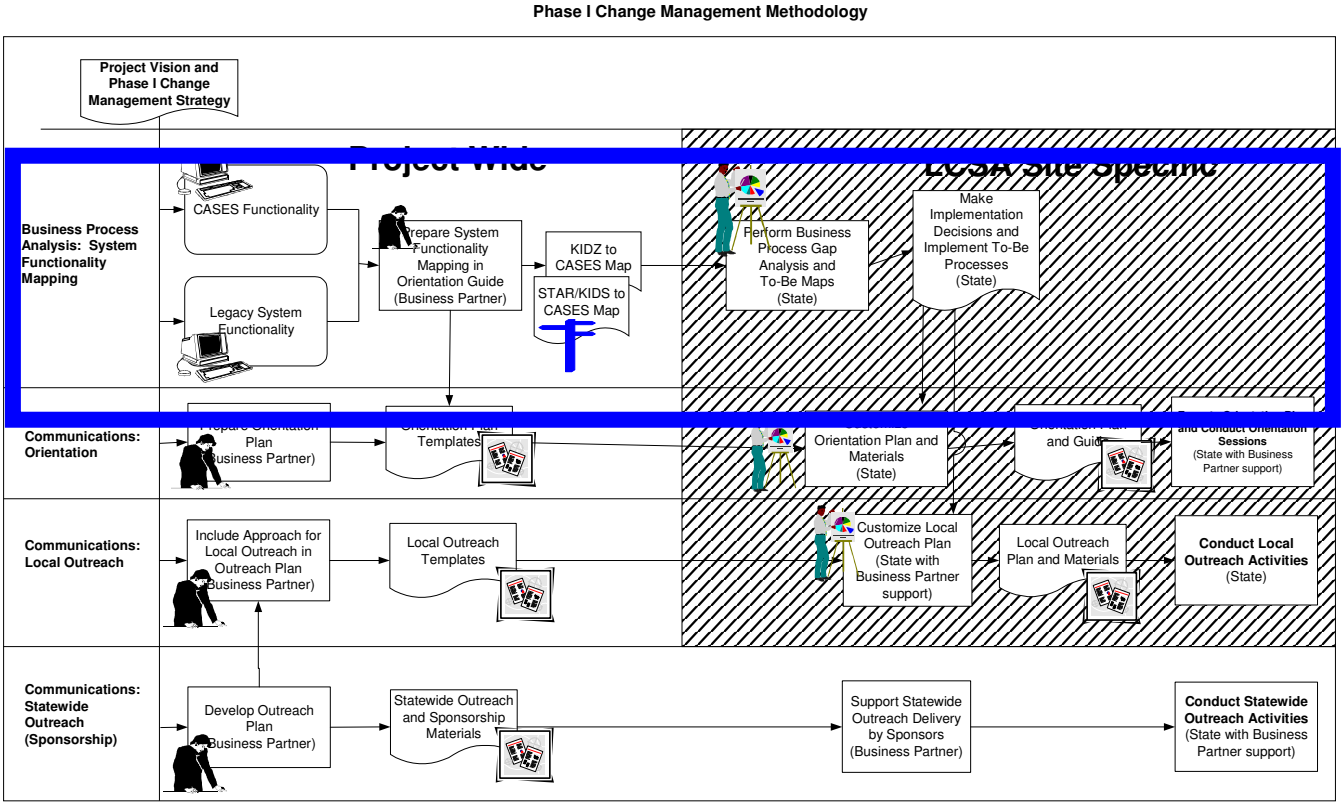


Figure 3C.13.2.2-6 Phase I Change Management Strategy - *Phase I change management strategy includes business process analysis system mapping, which is highlighted above.*

Phase I Business Process Analysis activities focus on conducting a functionality gap analysis between CASES and the LCSA legacy system. To accomplish this, the alliance provides the LCSA counties with a mapping of the CASES functionality to their existing system. The State then determines how to transition CASES functionality into their operations, policy and procedures, and documentation. We believe this is the appropriate approach to Phase I business process analysis activities because:

- LCSAs currently using CASES have used a similar approach to successfully make the transition to CASES
- It allows LCSAs to use their site-specific operational expertise to implement CASES with minimal disruption to business operations

System Functionality Mapping

Alliance staff lead the effort, working with LCSA staff from the lead consortia counties, to map CASES system functionality to one of their two safe-haven consortia systems (KIDZ, Star/Kids). These system functionality maps are documented in the Orientation Guide (TM 055). Each map will include the following information:

- **Business Process Area**

  Describes the processing area supported by the functionality. "Establishment" is an example of a business process area.

- **Legacy System Functionality**

  Describes the scope of legacy system functionality supporting the business process area.

- **CASES Functionality**

  Describes the scope of CASES functionality supporting the same business process area.

- **Comments**

  Documents details regarding any significant differences between CASES and the legacy system (scope of functionality or how a function is performed).

State staff use this information as a key input to the gap analysis activity. The purpose of the gap analysis is to determine what decisions must be made to accommodate the LCSA transition to CASES. Once alliance staff has provided the functionality maps, State staff is responsible for leading the analysis and transition effort. Alliance staff will be available for guidance and support throughout the analysis and transition activities.

Phase I Communications: Orientation

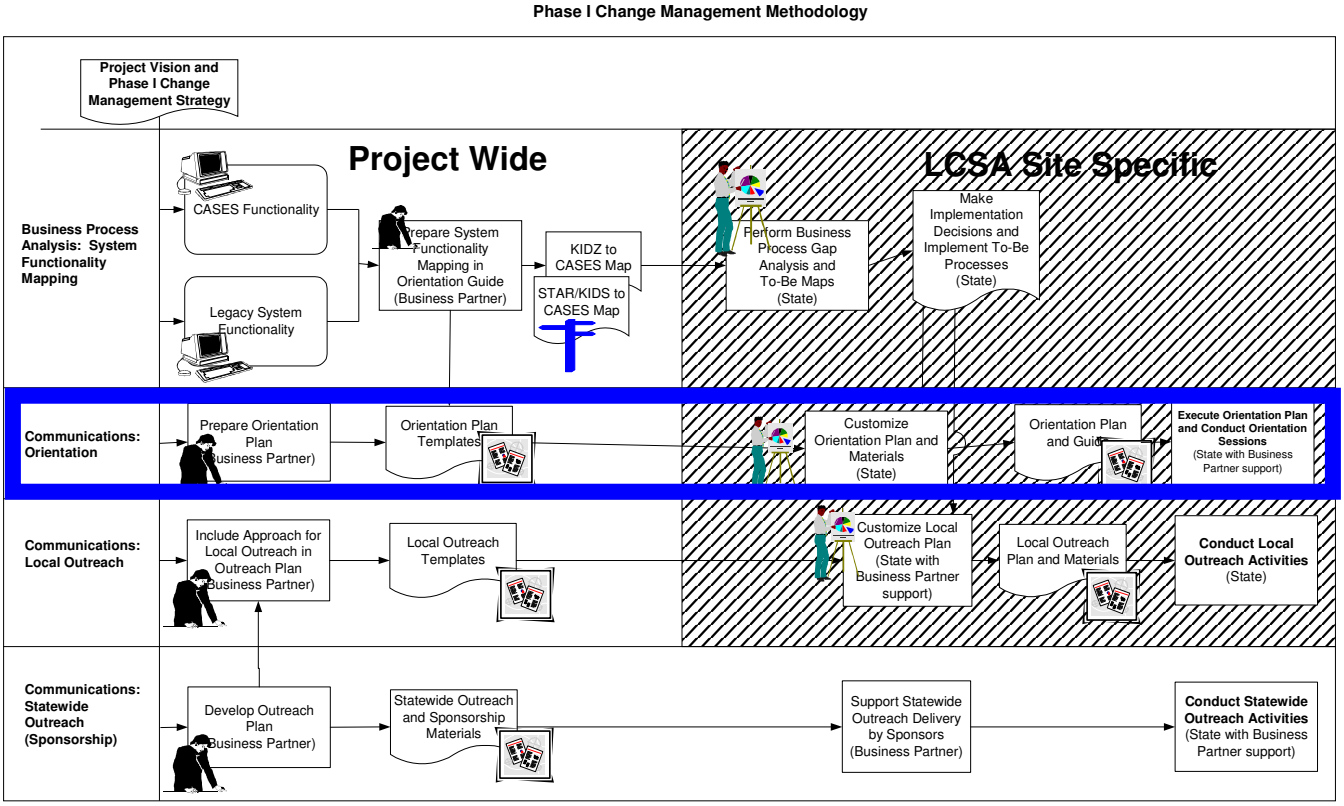


Figure 3C.13.2.2-7 Phase I Change Management - Strategy - *Phase I change management strategy includes orientation of new users, which is highlighted above.*

Phase I orientation activities are conducted in the 14 counties transitioning to CASES. The baseline Orientation Plan describes the change management planning processes for LCSAs to transition from the legacy system to CASES and for the DCSS and the LCSAs to transition to Version 1 of the CSE system. Based on the baseline Orientation Plan developed by the alliance, county personnel will use the Orientation Guide, which describes the mapping of the system functionality, to apply it to their operations. The Orientation Plan and Guide also provide templates of orientation and communications materials to the counties, who then customize the materials with implementation decisions before conducting orientation sessions for users. Alliance staff will support the State in conducting the tailored orientation sessions for each county.

Development of Baseline Orientation Plan (CDL TM 054)

Using information from sponsors, the alliance develops the baseline Orientation Plan (CDL TM 054). The baseline Orientation Plan serves as the roadmap for the customization and delivery of information to State and LCSA users.

The Orientation Plan, once customized for a specific LCSA site by the State, describes the messages, delivery mechanisms and timing for internal personnel, including:

- Methodology
- Activities

- Roles and responsibilities
- Development approach and use of the Orientation Guide
- Communications plan

The communications plan contains:

- Identification of internal audience groups
- Key Messages
- Delivery methods
- Timeframe for delivering information
- Feedback mechanisms
- Exit criteria

Development of Orientation Guide (CDL TM 055)

After the approval of the Orientation Plan, the alliance develops the baseline Orientation Guide. The Orientation Guide includes the mapping of current system functionality to CASES functionality, and the Orientation Plan contains the procedures, instructions and materials for conducting county-specific orientation activities. Both the Plan and the Guide include templates that can be applied by State staff to produce county-specific orientation sessions, written communications and meetings.

Delivery of Local Orientation

Based on the Orientation Plan and the Orientation Guide, State staff customizes the orientation materials to meet the requirements of the specific county. The alliance supports county personnel in conducting the new system orientation sessions prior to system training. The orientation sessions are used to describe the reasons for change, program improvements, local transition activities, training methods, and the timeline for implementation.

Phase I Communications: Outreach and Sponsorship

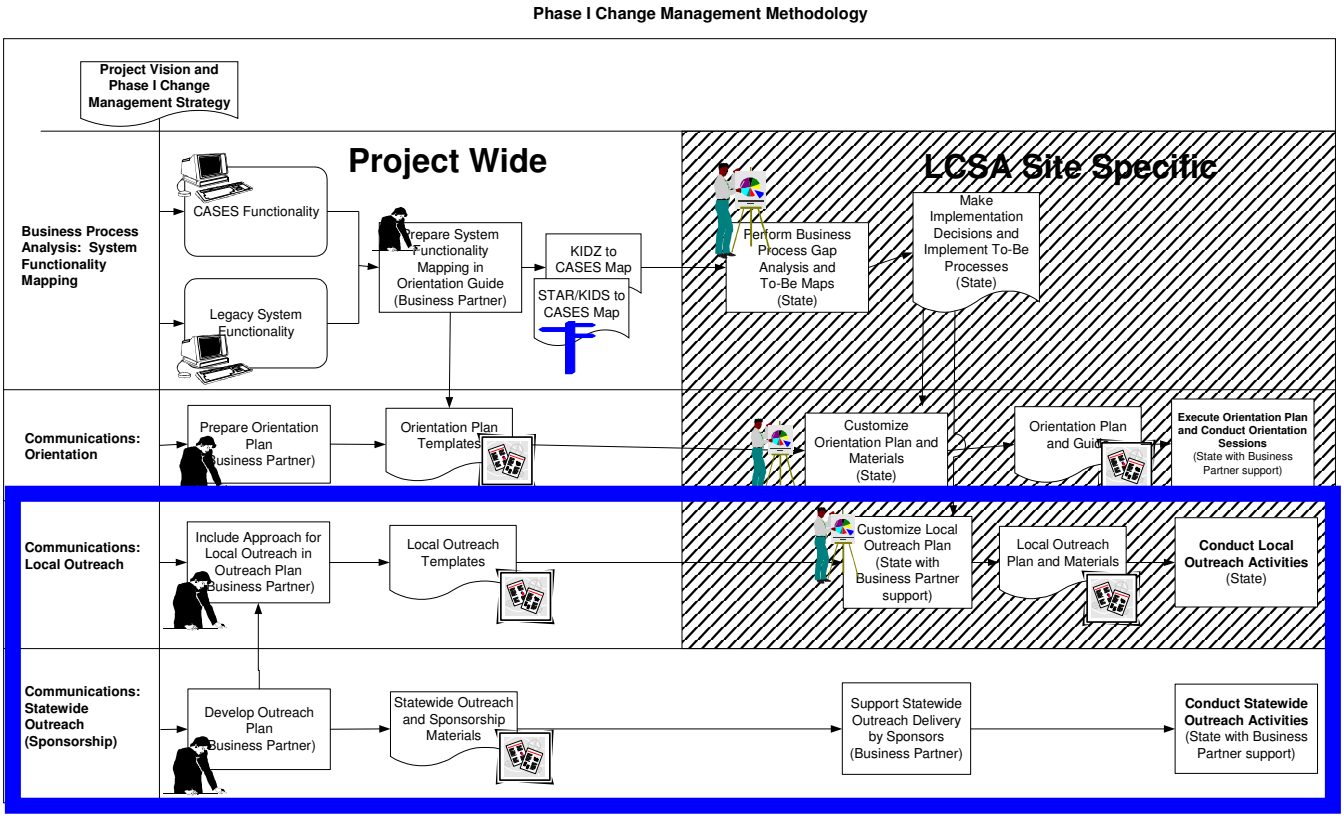


Figure 3C.13.2.2-8 Phase I Change Management Strategy - *Phase I change management strategy includes local and statewide outreach, which is highlighted above.*

Phase I involves two levels of outreach. At the LCSA level, local external entities, including customers in the 14 counties converting to CASES, will receive information from the State staff specific to their transition activities.

At the statewide level, the alliance team will prepare an Outreach Plan that facilitates distribution of general information about the CCSAS CSE project. The State will distribute statewide information to make external entities in all counties aware of general project status and progress.

Development of the Outreach Plan (CDL TM 053)

Using information from the stakeholder analysis being developed by the DCSS, the alliance develops the Outreach Plan with input from the State staff on the Communications and Outreach Working Group. The Outreach Plan serves as a roadmap for the development and delivery of messages to external entities across the state, including customers of the child support program. As a part of developing the Outreach Plan, the alliance team identifies the concerns, expectations and needs of key external entities as well as potential benefits each of these groups will receive as a result of the project.

The Outreach Plan contains:

- Identification of external audiences
- Development of messages

- Distribution and delivery methods
- Timeframe for delivering information
- Assessment/feedback mechanisms
- Exit criteria

The Outreach Plan documents the approach for communicating with external entities, including outreach templates (e.g., newsletter, web site information).

Delivery of Local Outreach

Based on the Outreach Plan, the alliance team provides templates of outreach materials for the LCSAs. The alliance also provides samples of customized materials from previously implemented counties. State staff is responsible for developing county-specific materials as well as delivery and distribution of communications to local external entities. The alliance is available for guidance and support for outreach activities.

Delivery of Statewide Outreach

In the area of statewide outreach, the alliance works with CCSAS project staff and the State's Communication and Outreach Working Group to establish a process for approval of information to be distributed statewide. As part of developing the Outreach Plan, alliance staff prepare templates and samples for communications pieces. CCSAS project and DCSS staff participate in the drafting of content as well as review and approve communications pieces prior to release.

Statewide information includes project goals, scope, and benefits of the implementation approach. The alliance will support program and project sponsors, including CCSAS Project leadership, DCSS leadership, and the Communication and Outreach Working Group, in their delivery of project-related information to external entities. The delivery of these messages by key sponsors is essential to gaining audience buy-in and a successful transition.

3C.13.2.2.2.2 Phase II Specifics

The Change Management objectives of Phase II are to:

- Facilitate the transition of 58 counties to the CCSAS CSE Version 2 application
- Orient LCSA and DCSS system users to changes associated with the Phase II initiative
- Inform external entities of changes that will impact their operations or processes used to interact with the CSE program through outreach initiatives

Figure 3C.13.2.2-9 presents an overview of the Phase II Change Management Strategy.

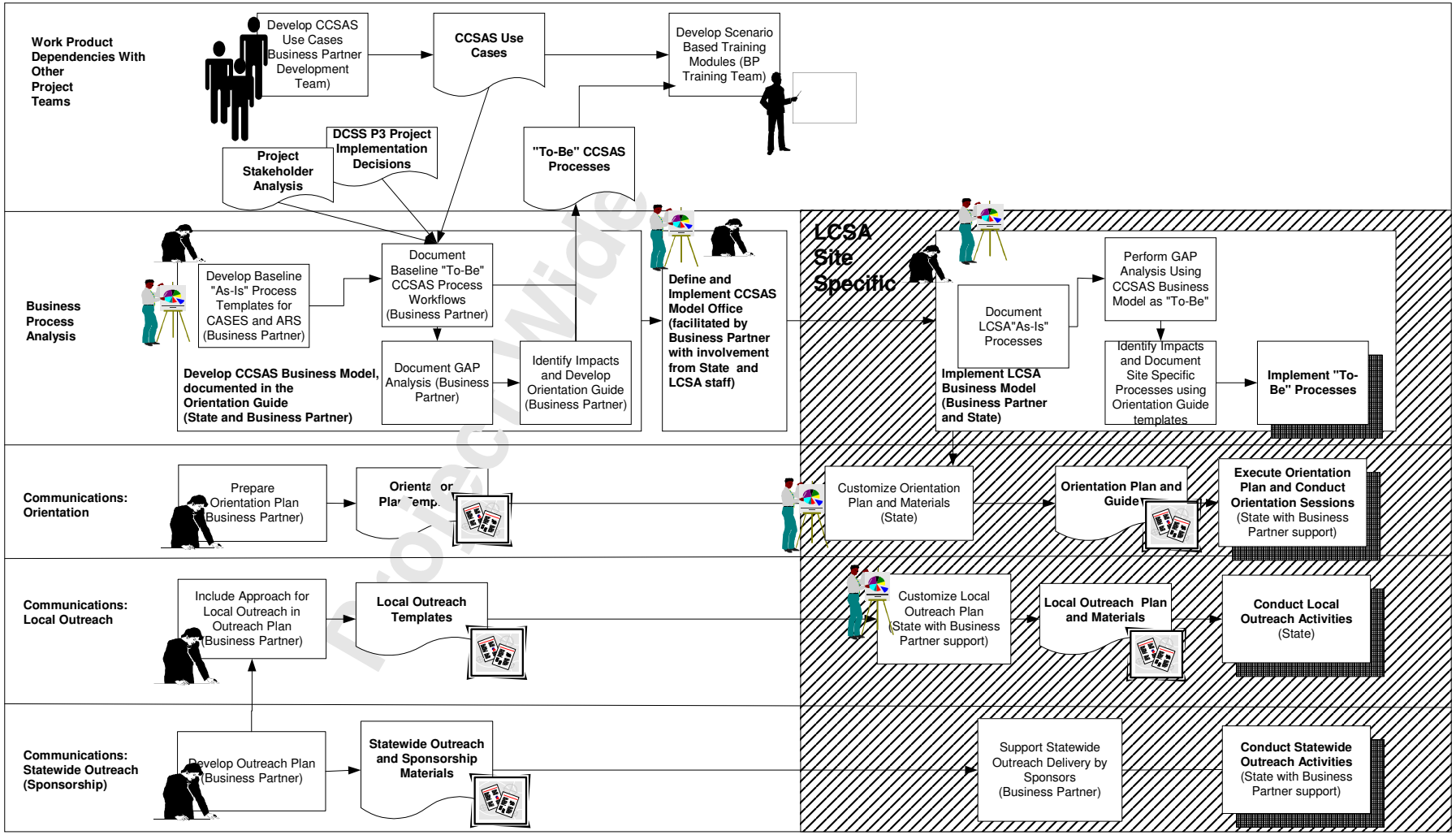


Figure 3C.13.2.2-9 Overview of the Phase II Change Management Strategy – *This methodology provides insight to all Phase II Change Management activities to facilitate communication across user communities.*

Phase II Business Process Analysis

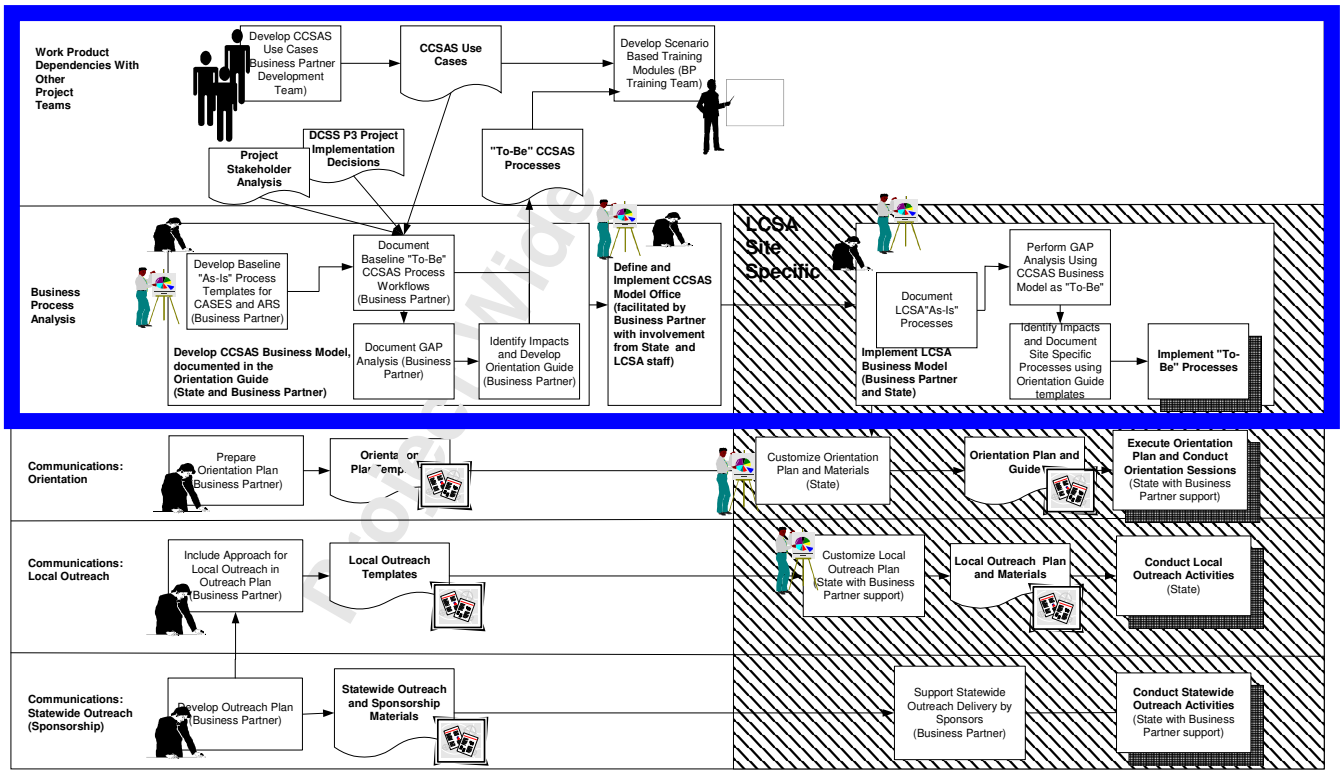


Figure 3C.13.2.2-10 Phase II Change Management Strategy - *Phase II change management strategy includes business process analysis, which is highlighted above.*

The Phase II business process analysis initiative facilitates the definition of and transition to uniform business processes within the statewide Child Support Program across the state. During Phase II, the alliance conducts the As-Is, To-Be, gap analysis and impact assessments, which are documented in the Orientation Guide (TM 055). These activities culminate in the development of the CCSAS Business Model.

The CCSAS Business Model is used to drive the implementation of uniform and standardized child support processes in the LCSA sites. The State leads the effort to implement the CCSAS Business Model in their locations, maintaining the authority to determine the organizational structure and job responsibilities of their sites.

We believe this is the most appropriate approach because:

- Business process uniformity is a key goal of Phase II of the CCSAS CSE project. A defined uniform model provides the vision and information needed for the LCSAs to make progress toward this goal.
- A model, templates and defined approach provide the LCSAs with the structured methodology and information needed to determine how to implement standardized processes in their sites.

- LCSA involvement and decision making in the business process analyses is vital to obtaining staff buy-in and acceptance.
-

Business Process Analysis Scope and Methodology

The business process analysis consists of the following activities:

- Definition of the CCSAS Business Model
- Implementation of the CCSAS CSE Model Office
- Business Process Activities of Each County

(a) Definition of the CCSAS Business Model

The alliance reviews the system requirements as defined in the SCP and P3 Project Implementation Decisions provided by the State as a key input to the development of the CCSAS Business Model. In addition, the CCSAS CSE Project Application Development Team provides the change management team with the CCSAS CSE Use Cases. A Use Case defines how the new CCSAS CSE Version 2 application will function within the context of a specific CSE business process. This information is vital to the development of business process flows and training modules.

In creating the CCSAS Business Model, the alliance also participates in the State's business process standardization activities by reviewing the business practices, policies, procedures, and regulations information. As appropriate, the alliance will participate in workgroups led by the State to determine the CCSAS Business Model.

The CCSAS Business Model will consist of the 12 major child support business processes:

- Case Initiation and Update
- Establishment
- Enforcement
- Locate
- Obligation Management
- Collections
- Distribution
- Disbursement
- Document Generation
- Report Writing
- Program Monitoring
- Customer Service

The development of the CCSAS Business Model occurs through the following activities:

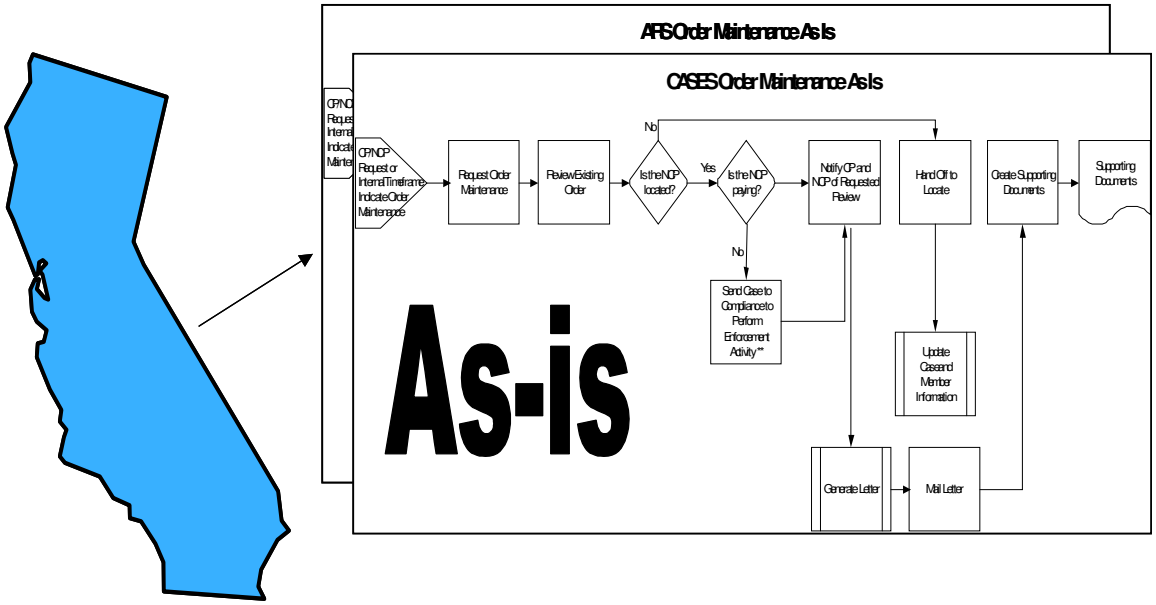
- Create As-Is business processes to document the current workflow under CASES and ARS
- Develop To-Be business processes to define the proposed workflow under the new CCSAS CSE system

- Perform gap analysis to determine the differences between current and future state
- Perform impact assessment to determine the actions necessary for the key areas of change

Using inputs such as existing "As-Is" process maps and the system functionality mapping completed in Phase I, the alliance team works with the State to develop the As-Is business process templates for CASES and ARS. These As-Is templates processes are generic and not specific to a particular county. The As-Is process templates are activity-level process maps depicting the flow between activities, decisions and documentation. Business process flows are created at the activity-level, not the task or step level. Activity level workflows illustrate the flow and relationship between activities, decisions and documentation. Review of business practices, policies, and procedures will be based upon the State baseline established for the child support enforcement program.

Figure 3C.13.2.2-11 is an example of an As-Is process map.

Figure 3C.13.2.2-11 Phase II As-Is Analysis: *One aspect of the business process analysis is to develop the As- Is*



models that document current processes under CASES and ARS.

The As-Is process maps act as the baseline picture of the current state in each LCSA. From this baseline, we will work with State staff to determine the differences between the current LCSA processes and the CCSAS Business Model processes.

To create the To-Be processes, the change management team works with a core group of CCSAS project, DCSS and LSCA staff designated as the CCSAS Business Model Workgroup. The alliance change management team participates in this workgroup. The role of the CCSAS Business Model Workgroup is to identify issues, make recommendations, determine documentation usage, and confirm decisions. It will be

important that a key State representative be assigned responsibility to manage this process and gain approval of the CCSAS Business Model in a timely manner, as Phase II implementation is dependent on this model.

Key inputs to the workshops include the State's P3 business process standardization decisions and best practices information, the system requirements, the CCSAS CSE application design (Use Cases), and the As-Is Process Maps. See Figure 3C.13.2.2-12 for an example of a To-Be Process Map.

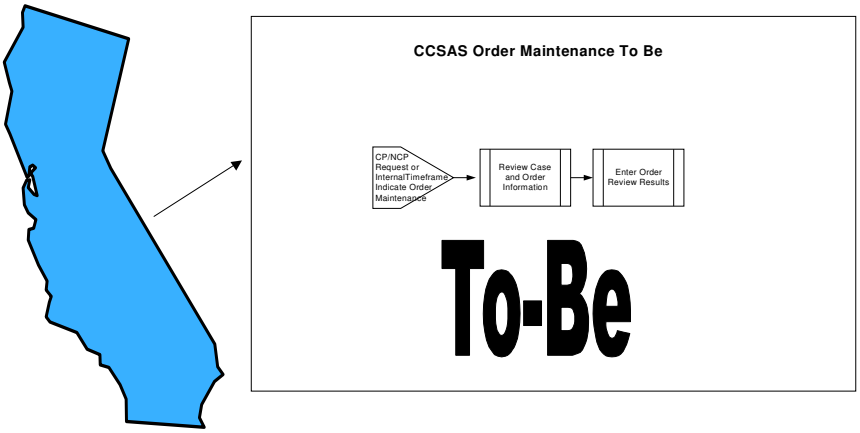


Figure 3C.13.2.2-12 Phase II To-Be. *One aspect of the business process analysis is to develop the To-Be model that documents processes under the CCSAS CSE Version2 system.*

During the workshop sessions, the alliance facilitates a process that helps the Workgroup conduct a gap analysis and identify the differences between the As-Is and To-Be processes. The impact assessment generates the recommendations and potential action items for transitioning the state toward greater process uniformity.

The ultimate result of the Workgroup sessions is the CCSAS Business Model. The CCSAS Business Model represents the future CSE Program uniform business processes to be implemented at the LCSA sites. The To-Be business process map templates are included in the Orientation Guide (TM 055) and used as input to the orientation, outreach and training materials. The Orientation Guide captures the CCSAS Business Model information and maps the old business functions with the future business processes.

Implementation of the CCSAS CSE Model Office

The alliance implements a Model Office based upon the CCSAS Business Model. The purpose of the Model Office is to demonstrate how the To-Be operational environment functions with the new CCSAS CSE application.

Implementation Strategy	Model Office Strategy
Enhance user acceptance of the system	Provides users a safe environment to test unrehearsed business scenarios
Reduce user learning curve	Allows users to become familiar with the system prior to training
Demonstrate future operations	Provides a safe environment that simulates future business process flow
Demonstrate integration	Provides an environment that demonstrates new documentation aligned with functionality
Verify software supports business needs	Users can see how they accomplish their job assignments

Figure 3C.13.2.2-13 CCSAS CSE Model Office - *The Model Office simulates the future CCSAS CSE environment..*

The Model Office located at the project site demonstrates the interplay of the new CCSAS CSE system and operational workflows to DCSS staff and CCSAS project staff as well as LCSA personnel during user orientation sessions. The Model Office consists of a separate application environment with a copy of the CCSAS CSE application and a single database environment that will be refreshed through a batch process. To minimize Model Office environment maintenance costs and minimize confusion with the e-learning, conversion data acceptance testing, and application testing environments and processes, the Model Office does not integrate with other environments.

Business Process Analysis Activities at Each County

Once the CCSAS Business Model is published in the Orientation Guide and made available statewide, counties use the information to determine and implement changes to reach the “To-Be” state at each county. The alliance supports the counties in the process review, gap analysis, detailed impact assessment, generation of action items, and execution of the action items for the transition to the CCSAS Business Model.

Using the CCSAS Business Model, the State customizes the As-Is process templates to reflect their county-specific processes, determine the implementation of the To-Be processes for their county, and conduct the gap analysis to determine what changes must be made to workflow, roles, and responsibilities. Figure 3C.13.2.2-14 illustrates an example item from an impact assessment action log.

Business Process	Impact Type	Impact	Action	Resp.	Due Date and Status
Case Initiation	Role	With CCSAS CSE, data entry personnel no longer need to enter information into three different systems. Currently two staff members spend 20% of their time performing this task.	Determine appropriate replacement tasks for time savings.	LCSA Office Manager	10/01/20XX IP

Figure 3C.13.2.2-14 Impact Assessment Action Log - *Comprehensive action log documents the necessary actions required to ready the site for implementation.*

A documentation analysis is also performed in order to identify required changes to forms. The alliance and State personnel analyze existing documents, forms and reports, compare against a list of the CCSAS CSE future documentation and determine the disposition of each document.

Document	Produced by Current System	Produced by Future System	Action
Document 123	Y	Y	None
Document 234	N	Y	New form, determine requirements and usage.
Form 123-01	Y	N	Obtain permission to eliminate form or develop plan to produce external to the new system.

Figure 3C.13.2.2-15 Document Analysis - *Document disposition is performed.*

Outputs from the Business Process Analysis

The business process analysis activity drives the LCSA implementation decisions. It also contributes to the work of the other implementation disciplines:

- **Training:** The To-Be analyses in the CCSAS Business Model serve as an input to the development of scenario-based training curricula. The business processes provide a framework for the flow of activities, decision points and documentation associated with the training modules.
- **Communications:** As more counties transition to their new system, the alliance is better able to share examples of benefits and goals the system has achieved. With such information, the user orientations better focus on implementation benefits from a user perspective. Also, sharing other county successes and ways in which the system has met agency workflow objectives helps new counties visualize how the system can most effectively enhance the processes in their agency.
- **User Support**: Business process analysis prepares teams to effectively conduct post-cutover user support by becoming familiar with how the LCSA intends to move toward uniform business processes and integrate the system into its daily operations.

- **Site Set-up:** Information from the business process analysis assists in readying the site for implementation. For example, information about the roles and responsibilities of LCSA personnel resulting from the analysis is used to help define security authorizations.

The business process activities for a given LCSA produce valuable information used in preparing LCSA staff for the cutover to the new system. Valuable information is gathered that prepares the CCSAS CSE application to operate appropriately within the LCSA. Additionally, the business process analysis results in the ability to perform site setup in advance of cutover, limiting disruption to users during the transition to the new system. Site setup includes:

- Gathering site-specific data required for an LCSA to successfully cut over to a new system
- Managing the entry of this data for the new system
- Validating these data

Examples of site-specific data are security profiles, user data, and LCSA county parameters (e.g., addresses and court information). The alliance site setup approach also performs a number of system validation checks prior to cutover, including the validation of each user's security/log in and workflow configuration on the new system. These validation checks occur prior to the system's first use in the site so that users can begin using the new system immediately after cutover.

Phase II Communications: Orientation

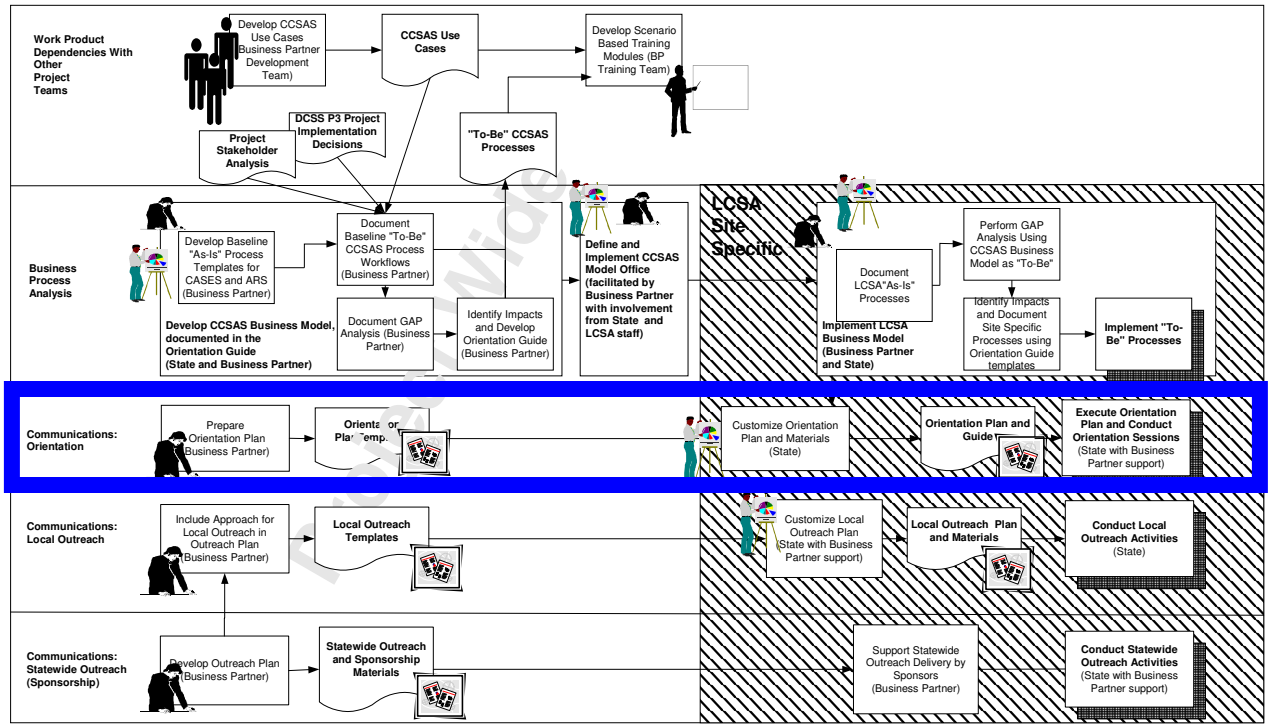


Figure 3C.13.2.2-16 Phase II Change Management Strategy - *Phase II change management strategy includes orientation of users, which is highlighted above.*

During Phase II, orientation and internal communications to those being transitioned to the new system occur to support the transition to the CCSAS CSE application. The alliance updates the baseline Orientation Plan, and communications materials already developed for Phase I to reflect the requirements of Phase II. The counties customize, produce, and deliver and/or distribute the information. The alliance supports the counties in conducting the orientation sessions, in particular as alliance staff will have the most familiarity with the CCSAS CSE system processes.

Update the Baseline Orientation Plan (CDL TM 054)

During Phase II, the alliance updates the baseline Orientation Plan with the messages, delivery mechanisms and exit criteria for communicating CCSAS CSE information to users. This plan will include information about how the new CCSAS CSE application affects their work, jobs, and daily tasks is important for DCSS and LCSA staff to understand. In addition, administration and management information will be provided, including:

- Planned improvements to service and staff productivity
- New reporting capabilities
- Functions and features providing them with better management and decision-making tools
- Implementation strategy
- Timelines for activities and operation disruptions
- Staff resource requirements

Staff need to understand how the new CCSAS CSE systems may affect them, including information on:

- New roles and responsibilities
- How the system will help them better do their jobs
- Overall benefits to the performance of the CSE program
- Benefits of the new system from a user perspective

The alliance “straw man” model presented below serves as a starting point for identifying target audiences and delivery methods. Along with the results of the DCSS’ stakeholder analysis, the information contained in Figure 3C.13.2.2-17 provides input to the baseline Orientation Plan.

Audience	Objectives	Delivery Methods
DCSS and LCSA Users	• Facilitate workers' understanding of the benefits of CCSAS CSE as they pertain to them • Familiarize workers with the business models • Provide channels for feedback and questions • Involve them in shaping the change	• CCSAS CSE project web site • Paper-based memos • Newsletters • Email • Meetings
DCSS and LCSA Managers / Supervisors	• Facilitate understanding of the CCSAS CSE benefits as they pertain to them and their staff • Familiarize staff with the business models • Provide channels for feedback and questions • Involve them in shaping the change • Participate in shaping the implementation •	• CCSAS CSE project web site • Paper-based memos • Newsletters • Email • Readiness surveys • Meetings

Figure 3C.13.2.2-17 Strawman Model for Orientation Plan - *The Orientation Plan provides information to users..*

Update of Orientation Guide (CDL TM 055)

For Phase II, the CCSAS Business Model provides key content for the update of the Phase I Orientation Guide Template to reflect Phase II information.

Delivery of Local Orientation

During the site readiness step, orientation information is communicated to LCSA staff via written communications, in meetings or during orientation sessions. During the pre-cutover step, the alliance supports the State staff in conducting the new system orientation sessions prior to system training. The orientation sessions are used to describe the reasons for change, program improvements, local transition activities, training methods, and the timeline for implementation. The session uses the CCSAS Business Model information to familiarize staff with the future business processes, job activities and functionality of the new CSE environment.

The role of alliance staff is to monitor site communications and orientation activities. Using the exit criteria as part of the Pre-Implementation Readiness Assessment, the team determines if communications Readiness Metrics have been met.

Phase II Communications: Outreach

During Phase II, the alliance updates the baseline Outreach Plan and materials to reflect a more thorough statewide and local outreach effort required for transition to the CCSAS CSE system. The alliance staff provides the outreach materials to the State and supports the State in preparing for and conducting local outreach activities.

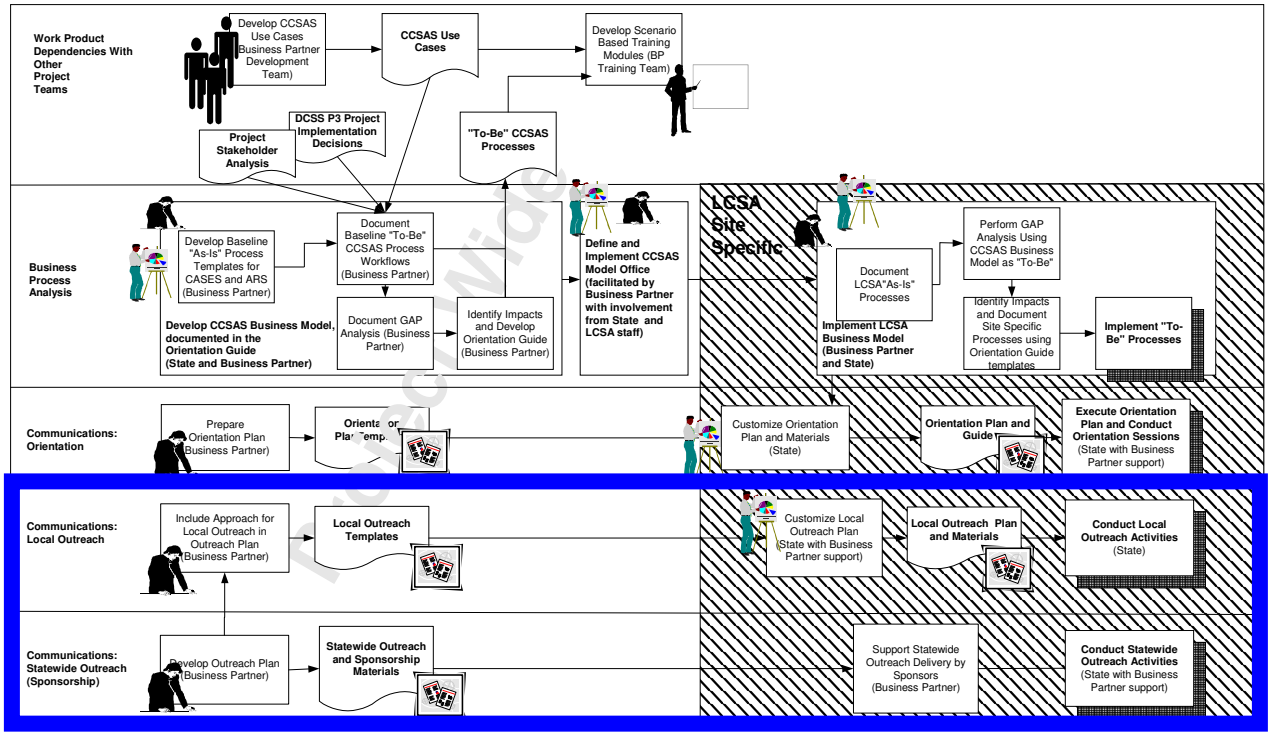


Figure 3C.13.2.2-18 Phase II Change Management Strategy - *Phase II change management strategy includes local and statewide outreach, which is highlighted above.*

Update of Baseline Outreach Plan (CDL TM 053)

The alliance team updates the baseline Outreach Plan with information regarding how the new statewide system benefits external entities. The baseline Outreach Plan includes outreach material templates of key messages and delivery methods for notifying customers of:

- Changes in member and case information
- Where to send payments
- Contact information
- How to access the new system

Our experience with child support has allowed us to develop a "straw man" model, as shown in the following table, which outlines the objectives to be achieved with each target audience. With the DCSS's stakeholder analysis, this acts as a starting point for creating the Outreach Plan.

Audience	Objectives	Delivery Methods
Custodial Parents and Non-Custodial Parents	• Communicate changes and schedules that effect the way parents interact with counties and the state • Indicate where to send payments • Provide contact information for questions	• DCSS web site • Customer Self-Service web site • Paper-based letters • Press releases
California State Legislature	• Report and demonstrate progress of CCSAS CSE • Address concerns • Maintain open lines of communications	• Paper-based memos • Topic-specific correspondence • One-to-one update in-person meetings
County Boards of Supervisors	• Maintain open lines of communications • Address concerns	• DCSS web site • Paper-based memos • Topic-specific correspondence • Presentations • Meetings • Conferences
District Attorneys	• Address concerns • Maintain open lines of communications	• DCSS web site • Paper-based memos • Topic-specific correspondence • Meetings • Conferences
Unions	• Facilitate unions understanding of the benefits of CCSAS CSE as they pertain to them • Familiarize workers with the business models • Provide channels for feedback and	• DCSS web site • Paper-based memos • Road shows • Meetings • Conferences

Audience	Objectives	Delivery Methods
	questions • Involve them in shaping the change	
Media	• Maintain open lines of communications	• DCSS web site • Press releases
Employers	• Provide channels for feedback and questions • Involve them in shaping the change • Communicate benefits as they pertain to them	• DCSS web site • Press releases • Topic-specific correspondence
Financial Institutions	• Provide channels for feedback and questions • Involve them in shaping the change • Communicate benefits as they pertain to them	• DCSS web site • Press releases • Topic-specific correspondence
National and State Child Support Organizations	• Communicate changes and schedules that affect the way parents interact with counties and the state	• DCSS web site • Paper-based memos • Conferences

Figure 3C.13.2.2-19 Strawman Model for Outreach Plan - *The Outreach Plan provides specific information to external entities.*

Delivery of Local Outreach

During the readiness step, project staff supports the local LCSA staff to review the local Outreach Plan template, identify county-specific external entities, and customize the materials.

Delivery of Statewide Outreach

During Phase II, the delivery of the statewide messages may take multiple forms as depicted in Figure 3C.13.2.2-19. It may also include the preparation and distribution of materials, which include brochures or fact sheets via electronic or paper media.

3C.13.2.2.3 [RESERVE]

3C.13.2.3 Implementation Management

Implementation Management as a discipline provides the organizational framework for the implementation team and the approach for managing each implementation.

The alliance approach for managing CCSAS CSE implementation activities is structured into two components: 1) managing overall implementation activities statewide; 2) managing implementation activities in each county. This section describes the Implementation Organization, which is structured to provide an effective framework for managing the overall implementation activities statewide. Statewide Implementation Administration and Supervision manages how the overall team is organized to execute implementation activities, including a definition of roles and responsibilities. Site Implementation Management conducts planning and coordination activities for each LCSA implementation. Implementation Resource Groups provide resources with required expertise to the Site Implementation Teams. Together, the Site Implementation Management Group and the Implementation Resource Groups oversee the work of the seven disciplines of implementation in each implementation county:

- Implementation Management
- Change Management
- Data Conversion
- Hardware/Software Installation
- User Training
- Local Interfaces
- On-Site Support

3C.13.2.3.1 The Implementation Organization

Since the approach to implementation management remains consistent throughout both phases of the project, the same organizational framework and site implementation approach remains in place throughout the project life. Figure 3C.13.2.3-1 presents the proposed Implementation Organization.

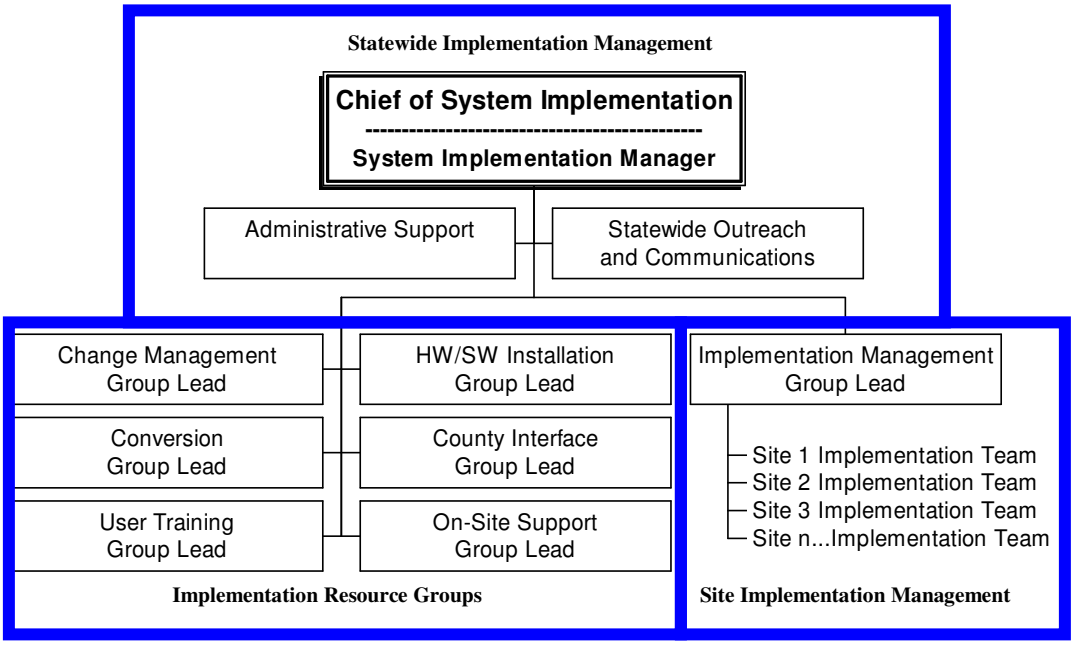


Figure 3C.13.2.3-1: The Implementation Organization - *The organizational framework and site implementation approach are consistent throughout the project life.*

Statewide Implementation Administration and Supervision

Statewide Implementation Administration and Supervision is the joint responsibility of the Chief of System Implementation and the System Implementation Manager, working in collaboration with the State's named project implementation executives. Statewide Implementation Administration and Supervision involves both the identification and management of integration points with the CCSAS CSE Project Application Development and Architecture Teams, management of both project wide and external communications, and project activity coordination with the executive management level staff of the LCSAs.

Site Implementation Management and Implementation Resource Groups

Site Implementation Management and Implementation Resource Groups are responsible for planning, managing and coordinating the work of staff within their single discipline area. They also contribute staff to the cross-discipline teams who work in the counties under the leadership of a Site Implementation Manager (one assigned to each LCSA), executing and supporting transition activities. For example, the Change Management Group Lead is responsible for defining the approach, activities and deliverables associated with executing the CCSAS CSE Project Change Management Strategy. Members of each alliance discipline group, such as the Change Management Group, are assigned to cross-discipline teams made up of staff from each of the six groups. This approach to maintaining a core group of highly qualified resources for each of the seven disciplines and then deploying them to cross-discipline teams at each site allows the alliance to establish teams of appropriate size and skill-set mix to meet site-specific requirements.

Role of CCSAS Project Staff within the Implementation Organization

The alliance recognizes the importance of staffing the Implementation Organization with both alliance and CCSAS project staff to achieve two objectives:

1. Leverage state staff expertise in child support operations
2. Build CCSAS project staff expertise in various implementation aspects of the project

The alliance understands that the CCSAS project staff members are available to support the implementation effort. We also understand that the CCSAS Project Transition Management Plan outlines some of the specific ways these staff will work with DCSS, LCSAs, and the alliance to coordinate the transition management activities. To this end, we propose that CCSAS Project staff work with the Site Implementation Management group to assist the Site Implementation Managers with work in the counties. We also propose that CCSAS Project staff work with the Change Management, Conversion, Training and On-Site Support groups. This allows for considerable input and high visibility of CCSAS project staff in the LCSA implementation efforts.

Additionally, the alliance will work with the Transition Management Steering Committee and the cross-functional Project Workgroups as mentioned in the CCSAS Project Transition Management Plan to coordinate implementation planning.

CCSAS Project Staff Role	Responsibility
Counterpart to the Chief of System Implementation	Represents the CCSAS project in coordination with the Chief of Implementation for the following activities: • Works with the CCSAS Project directors and other State leaders on a regular basis • Works with Project Executive and other project representatives on a daily basis • • Coordinates CCSAS LCSA Directors' Meetings
CCSAS Project Staff	Lead or participate in Project Workgroups Plan, execute, and evaluate activities in the areas of Change Management, Training, On-Site Support, and Communication and Outreach Conduct implementation work in counties as representatives of the CCSAS project
Transition Management Steering Committee	Responsible for directing and coordinating project transition activities with CSP restructuring and operations activities.
Project Workgroups (e.g., Training, Communications and Outreach)	Serve as SMEs and/or facilitate the input of SMEs from State, LCSAs, and others Develop issues and identify risks and develop strategies to resolve or mitigate these issues and risks Provide guidance, coordination, support and oversight of transition activities Identify issues and risks and mitigation actions

Figure 3C.13.2.3-2 Responsibilities of CCSAS Project Staff, Committees, and Workgroups *The alliance will work with the State staff through various methods as outlined in the CCSAS Transition Management Plan.*

3C.13.2.3.1.1 CCSAS CSE Implementation Roles and Responsibilities

This section defines the roles and responsibilities of the CCSAS CSE Implementation Organization, which includes:

- Statewide Implementation Administration and Supervision
- Implementation Resource Groups
- Site Implementation Management Group
- Site Implementation Teams

3C.13.2.3.1.1.1 Statewide Implementation Administration and Supervision

The Chief of System Implementation leads the alliance's Implementation Organization, assisted by a System Implementation Manager. Together, these two executive staff plan and coordinate the macro-level activities of the Implementation Discipline Groups, plus the Site Implementation Management Group, which together comprise the seven implementation disciplines.

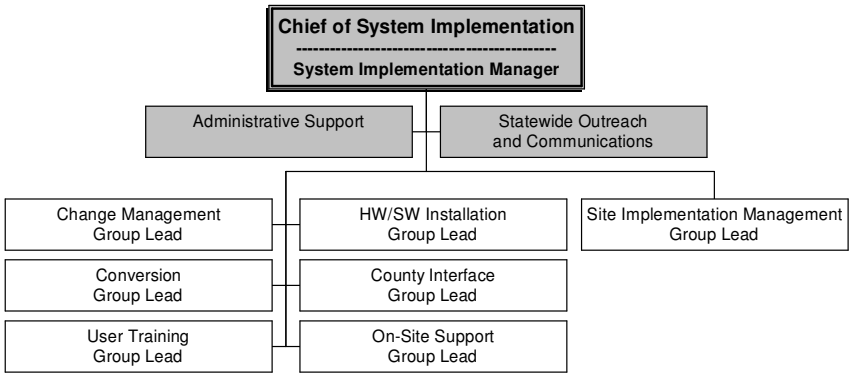


Figure 3C.13.2.3-3 Alliance Statewide Implementation Administration and Supervision *This shows the position of the alliance Chief of System Implementation and assistants within the organization.*

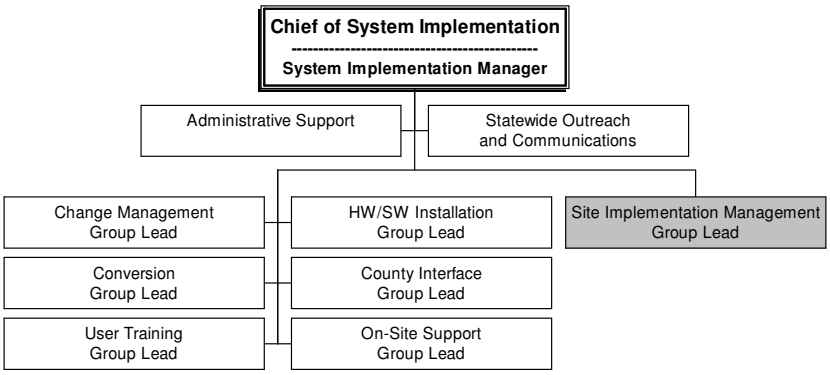
In coordination with CCSAS project staff, the Statewide Outreach and Communications Group supports communications activities that promote understanding about the CCSAS CSE project. This group works closely with sponsors and stakeholders on the statewide outreach activities. Figure 3C.13.2.3-4 describes the responsibilities of the Statewide Implementation Administration and Supervision:

Alliance Role	Responsibility
Chief of System Implementation	Assumes overall responsibility for the successful implementation of CCSAS CSE systems statewide Works with the Project Directors and other project representatives on a daily basis Participates in Executive Steering Committee Coordinates CCSAS CSE Directors' Meetings
System Implementation Manager	Has day-to-day management responsibility for the following teams of implementation resources: ▪ Implementation Management ▪ Change Management. ▪ Data Conversion ▪ HW/SW Installation ▪ User Training ▪ Local Interfaces ▪ On-Site Support Leads the development of materials, tools, and approaches for the implementation disciplines Determines resources to staff Site Implementation Teams Responsible for the implementation work plan, staffing, status reports, and scheduling
Administrative Support	Assists in producing implementation work plan and status reports Coordinates management processes and activities on behalf of the Chief of System Implementation and the System Implementation Manager
Statewide Outreach and Communications Group Lead	Develops the outreach and communications approach at both the State and local levels Works closely with CCSAS project staff and DCSS to perform statewide outreach activities Works closely with the change management discipline group to support the execution of outreach plans at the LCSA level Oversees the implementation-related sections of the CCSAS CSE project web site

Figure 3C.13.2.3-4 Alliance Roles and Responsibilities – *These represent the roles and responsibilities of the Statewide Implementation Administration and Supervision Group.*

3C.13.2.3.1.1.2 Site Implementation Management Group

Figure 3C.13.2.3-5 Site Implementation Management Group Lead - *This shows the position of the Site*



Implementation Management Group Lead within the organization.

The Site Implementation Management Group Lead is responsible for developing the approach and materials required to support the work of the Site Implementation Teams. The Group Lead has the following role and responsibilities:

Alliance Role	Responsibility
Site Implementation Management Group Lead	Leads the overall implementation effort in the LCSAs and DCSS Develops and maintains relationships with DCSS Regional Managers and LCSA Directors Coordinates LCSA Group Meetings Identifies and manages potential risks Manages the Site Implementation Managers and their corresponding Site Implementation Teams

Figure 3C.13.2.3-6 Roles and Responsibilities – *These represent the roles and responsibilities of the Site Implementation Management Group Lead.*

Each LCSA is assigned a cross-discipline Site Implementation Team comprised of alliance and CCSAS project staff. The Site Implementation Teams conduct and coordinate the activities required to prepare the county for transition. A Site Implementation Manager is assigned to each county as the single point of contact to work with the LCSA Director and the site's LCSA Site Coordinator (the LCSA staff person responsible for leading the transition effort in the LCSA). The Site Implementation Manager is responsible for coordinating the readiness activities and managing the Site Implementation Team.

Figure 3C.13.2.3-7 presents the structure for assigning members from each discipline group to an implementation team. Assignments are made based upon the implementation expertise and consortia system experience required by the site.

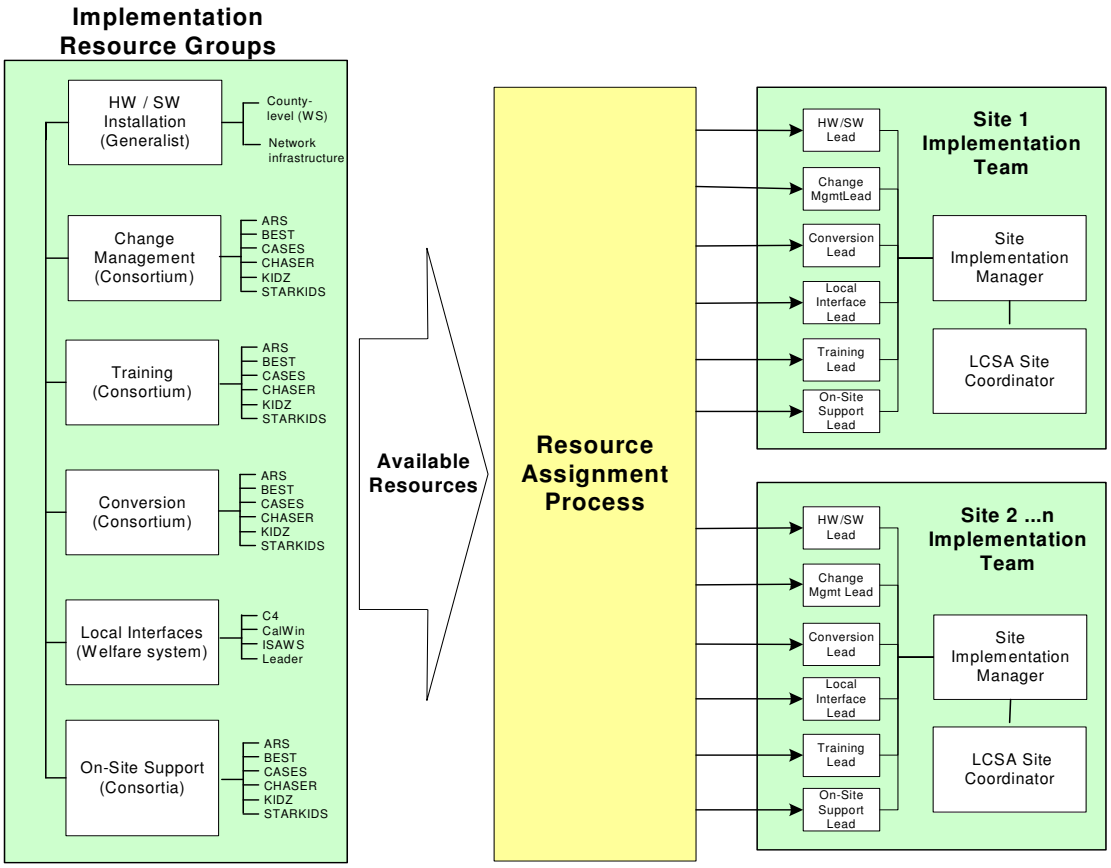


Figure 3C.13.2.3-7 The Alliance Staffing Approach for Site Implementation Teams - *Cross functional teams of alliance discipline staff are assigned to work with the Site Implementation Manager to implement each site.*

The alliance approach to staffing cross-discipline Site Implementation Teams provides a framework for an efficient allocation of our expertise and resources.

Site implementation teams assigned to an LCSA remain with that county throughout the entire transition process. This provides for continuity of staff that becomes knowledgeable in the operations of the particular LCSA. Some of the disciplines (i.e., conversion) will be organized to specialize by consortia system in order to leverage knowledge gained in previously transitioned counties within the same consortium. The number of resources assigned to each implementation team will depend on the LCSA's size and other readiness factors. With the exception of the critical months before and after transitions in some of the larger counties, it will not be necessary for most implementation team members to be present at the county site on a full-time basis. In very large counties, there will be several people assigned to the implementation management function to assist in coordinating the transition involving multiple county sites.

3C.13.2.3.1.1.3 Implementation Resource Groups

The Implementation Resource Groups are comprised of six groups of qualified resources offering expertise in the following areas:

1. Change Management
2. Data Conversion
3. HW/SW Installation
4. User Training
5. Local Interfaces
6. On-Site Support

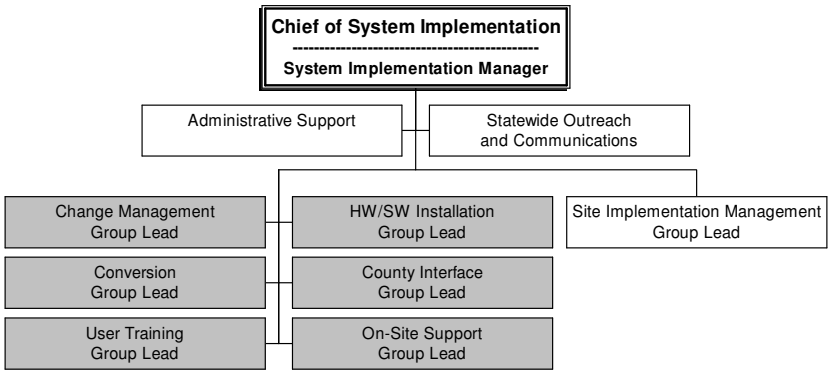


Figure 3C.13.2.3-8 Implementation Resource Groups - *The Implementation Resource Groups are comprised of six groups of qualified professionals.*

With the Site Implementation Management Group, these groups make up the implementation disciplines. A Group Lead manages each discipline group and is responsible for developing the approach and materials required to support the work of their respective discipline. Figure 3C.13.2.3-9 lists the general responsibilities of the Resource Group Leads. Although there may be some differences in the specific activities conducted by phase as outlined in Figure 3C.13.0-4, the following outlines general responsibilities:

Alliance Role	Responsibility
Change Management Resource Group Lead	Manages change management resources available to Site Implementation Teams Oversees change management activities that support LCSA and State-level implementations Coordinates development of LCSA communications and outreach plans Coordinates development of change management tools, deliverables, templates, and models Coordinates the business process analysis
Data Conversion Resource Group Lead	Manages conversion resources available to Site Implementation Teams Oversees conversion activities that support LCSA/site implementations Coordinates development of conversion tools, deliverables, templates, and models Works with consortia lead LCSAs to execute conversion activities

Alliance Role	Responsibility
HW/SW Install Resource Group Lead	Manages HW/SW roll-out resources available to Site Implementation Teams Oversees HW/SW roll-out activities that support LCSA/site implementations Coordinates development of HW/SW rollout deliverables, templates, and models
User Training Resource Group Lead	Manages training resources available to Site Implementation Teams Oversees training activities that support LCSA/site implementations, including delivery of classroom training and e –learning Coordinates development of training materials, deliverables, templates, and models Implements and maintains the Learning Management System
Local Interfaces Resource Group Lead	Manages county interface resources available to Site Implementation Teams Oversees county interface activities that support LCSA implementations Coordinates development of county interface deliverables, templates, and models
On-Site Support Resource Group Lead	Manages alliance on-site support resources available to Site Implementation Teams Oversees LCSA on-site support activities that support LCSA implementations Coordinates development of on-site support deliverables, templates, and models

Figure 3C.13.2.3-9 Roles and Responsibilities – *These represent the roles and responsibilities of the Implementation Resource Group Leads.*

3C.13.2.3.1.2 Staff Assimilation and Exchange Programs

The size, complexity, and length of the CCSAS CSE project will create a situation where new staff will need to be quickly and effectively oriented to the project, and staff knowledge of, and history with, the project will become an essential project asset. To properly orient new staff and to better leverage the knowledge base of experienced project staff during peak project periods, the alliance proposes two staff development programs:

- The Staff Assimilation Program
- The Staff Exchange Program

Staff Assimilation Program

Members of the implementation organization participate in an assimilation process to prepare them for their role on the project upon their joining of the project. Our approach accounts for assimilation of new staff in the initial stages of the project, as well as assimilation for new staff who are brought on the project in later stages as we ramp up staffing. Three assimilation modules are offered to members of the implementation organization in addition to the general project orientation that is offered to CCSAS CSE project members:

- Implementation-Specific Orientation
  Members of the implementation organization attend an implementation-specific orientation that is expected to be two days in length. This module introduces the CCSAS CSE implementation approach, templates and tools.
- Enhanced Training Session
  Members of the groups that work directly with CCSAS CSE users also attend an intensive training session on the application. This includes staff from the Implementation Management, Change Management and On-Site Support disciplines.
- Trainer Preparation Session
  Trainers attend a Trainer Preparation session in Phase II, which covers in-depth CCSAS CSE system training and instruction on training delivery.

Staff Exchange Program

The alliance Staff Exchange Program is aimed at achieving two objectives:

- Developing and leveraging staff project knowledge and expertise
- Validating that staff peak demands are effectively addressed

The alliance Staff Exchange Program provides staff with skills that allow management greater flexibility in making work assignments during peak implementation periods. The program does this by providing a means of early knowledge transfer, enhanced learning, and cross project training. The Staff Exchange Program operates by loaning staff with specific skills to other project teams for defined periods of time during the project lifecycle.

For example, members of the Implementation Organization may participate in system testing in order to gain in-depth knowledge of the application in preparation for rollout and on-site support. Similarly, members of the Development Team may be loaned to implementation teams in the LCSAs to provide in-depth application knowledge during the first days of implementing a new release. Members of the Help Desk Team additionally may be sent into the field to get more first-hand experience with user issues and questions prior to beginning their work on the Help Desk.

3C.13.2.3.2 Site Implementation Management

Implementation planning within each LCSA will be organized along distinct implementation planning stages and coordinated by the Site Implementation Manager to limit disruption to county operations. The alliance approach to planning is grounded in four assumptions:

- Readiness Metrics drive implementation planning
- Site Implementation Manager provides accountability
- Innovative tools facilitate implementation planning
- Regular review of lessons learned refines our approach

Readiness Metrics Drive Implementation Planning

Assessing readiness up front helps customize the site implementation team's activities to the needs of the individual site.

Readiness Metrics identify, by discipline, items for the implementation teams to address as they prepare to implement CASES and CCSAS CSE. Readiness Metrics are associated with key milestones across the six discipline areas of implementation. Readiness Metrics are reviewed and customized at the beginning of the planning effort to help participants understand the prerequisites and timelines for system implementation.

Site Implementation Managers track progress by working regularly with parties responsible for specific discipline groupings of the metrics. This process of consistent accountability provides a means for aligning expectations to current circumstances, monitoring progress in meeting each metric, mitigating overall implementation risk, and confirming and adjusting schedule assumptions.

Site Implementation Manager Provides Accountability

The Site Implementation Manager acts as the primary point of contact for LCSA staff, coordinating activities and team members from the various disciplines within the sites at each LCSA. We recognize that LCSAs are distinct organizations with different needs. To address county--specific needs, the Site Implementation Manager acts as the key relationship manager and point of contact for the county, working with the LCSA Site Coordinator to manage the implementation activities. As the key relationship manager, the Implementation Manager is responsible for identifying and resolving issues, identifying the most appropriate methods of conducting work/meetings in the county, and working to mitigate known risks.

Innovative Tools Facilitate Implementation Planning

The alliance approach includes innovative tools to support and coordinate the work for implementation in the LCSAs, including:

- The Implementation Planner – Supports implementation planning efforts by providing templates that promote the exchange of information between the alliance, DCSS, and LCSA staff.
- CCSAS CSE Project Web Site – Promotes communication throughout the CCSAS CSE community.

The Implementation Planner

The CCSAS CSE Implementation Planner serves as a roadmap for use by State, LCSA, and alliance staff while preparing to deploy CCSAS CSE in each LCSA. Specifically, The Implementation Planner provides a series of templates available through the CCSAS CSE project web site to help State staff perform readiness activities for each of the seven disciplines (e.g. conversion, training, etc.) Eventually, the completion of the templates in the Implementation Planner leads to the satisfaction of the Readiness Metrics. From there, the State, the local LCSA, and the alliance can make an informed decision regarding cutover readiness. Figure 3C.13.2.3-10 includes examples of some of the templates in the Implementation Planner.

Planner Section	Purpose
Pre-Assessment Survey	Provides baseline information on the LCSA organization to be completed by the LCSA during project initiation. Information is used to begin planning in each LCSA, such as the number of users and sites to be implemented.
LCSA Implementation Team Composition	Describes roles and responsibilities of the LCSA implementation team so that the LCSA can identify key individuals who would serve as the liaisons for the following discipline areas: Implementation Management Change Management Data Conversion Hardware/Software Installation User Training Local Interfaces On-Site Support The LCSA implementation team composition template estimates workload for each of the areas to guide the LCSA staff in making decisions on staff to select for participation in the LCSA implementation team.
LCSA Calendar/Workplan	Provides timeline for LCSA planning and rollout activities to guide schedule.
Readiness Metrics	Provides checklist of readiness criteria to be used jointly by project and LCSA staff to determine ability to deploy CCSAS CSE.

Figure 3C.13.2.3-10 Examples of Templates in the Implementation Planner - *The Implementation Planner includes templates for each discipline area that help prepare the site for transition. By completing the templates that support the Pre-Implementation Readiness Assessment and Readiness Metrics, the site has achieved readiness and is ready for cutover.*

The Implementation Center on the CCSAS CSE Project Web Site

The CCSAS CSE project web site provides an intranet site for child support staff in California. It includes general information about the CCSAS CSE project such as rollout schedules and newsletters, and will serve as a primary point of communication during the implementation cycle.

The CCSAS CSE project web site includes the Implementation Center, an area where information regarding the site implementations will be stored and shared. The Implementation Center is envisioned to include components dedicated to the following topics: reports, conversion, interfaces, implementation planning, production readiness, communications, lessons learned, and others as needed. Additionally, the Implementation Planner and the Readiness Metrics will be available on the CCSAS CSE project web site, allowing access to CCSAS project and LCSA staff. The Website will also include a link to the Learning Management System, which supports user training, scheduling, and registration.

Regular Review of Lessons Learned Refines Approach

In addition to already incorporating lessons learned from other projects, the alliance CCSAS CSE implementation approach allows for regular review of lessons learned. Our implementation approach assumes that refinements are made to the basic framework based on experiences throughout implementation. This “closed loop” process is especially important on implementation projects that take place over longer periods of time. During the course of the project, lessons learned will be reviewed and incorporated to strengthen our implementation approach.

The two-phased implementation approach exploits our ability to incorporate lessons learned by allowing the implementation approach to be validated using an existing system during Phase I. In preparing for Phase II, these lessons are incorporated in the review and revision of our implementation approach, where appropriate. Then, during Phase II implementation, the team can concentrate on testing the new CCSAS CSE system using a validated implementation approach.

Our plans for both Phase I and Phase II call for pilot implementations, as described in the rollout schedule in Section 3C.13.2.4. In addition to piloting the application during this period, we also look for opportunities to improve our implementation approach prior to moving into statewide rollout. To allow time to incorporate lessons learned during pilot, we schedule a period after completing pilot but prior to beginning statewide rollout that allows time to make adjustments in our implementation approach.

In implementations such as the statewide implementation of CCSAS CSE, there are ongoing opportunities to learn from the earlier LCSAs. The alliance implementation approach and materials are regularly reviewed to incorporate lessons learned from earlier CCSAS CSE implementation efforts based on the information gathered in the Post-Implementation Review Reports (CDL TM 039). The benefit of this approach is that it facilitates the incorporation of lessons learned into subsequent transitions.

3C.13.2.3.2.1 Implementation Planning

LCSA Implementation Planning occurs at two levels:

- Implementation Planning at Each LCSA
- Regional and Group Meetings

The following sections describe how the alliance conducts planning at each LCSA through the implementation steps and through Regional and Group Meetings.

Implementation Planning at Each LCSA

The alliance implementation methodology relies upon a joint effort between the State, LCSA staff, and the alliance to ready sites for transition. Prior to the start of planning in each LCSA, the alliance Site Implementation Manager works with the State to identify LCSA implementation team members. The State will identify a liaison for each of the discipline areas (see LCSA Liaison Responsibilities in Figure 3C.13.2.3-11). In some smaller LCSAs or sites, one individual could serve as the liaison for multiple discipline areas.

The implementation responsibilities of the Site Implementation Teams are summarized in Figure 3C.13.2.3-11. Although there may be some differences in the specific activities conducted by phase as outlined in Figure 3C.13.0-4, the following outlines general responsibilities:

Alliance Team Member Responsibilities	Site Implementation Team Role	LCSA Liaison Responsibilities
Provide each site with an **Site Implementation Manager** who: Acts as primary point of contact to LCSA Director, LCSA Coordinator, and other project sponsors in the LCSA Prepares site-specific work plan and coordinates the LCSA implementation process according to work plan Manages the implementation resources in each discipline for the LCSA Monitors aspects of readiness with LCSA Coordinator, with input from resources from each discipline Contributes to best practices/lessons learned body of knowledge after implementations Participates in the coordination of LCSA Group Meetings	Implementation Management	Designate an **LCSA Coordinator** who: Acts as primary point of contact to Site Implementation Manager and CCSAS project team Works with the Site Implementation Manager to coordinate the planning and execution of transition activities in the LCSA Manages the LCSA resources for each discipline working on CCSAS CSE implementation Monitors aspects of readiness with alliance Site Implementation Manager, with input from resources from each discipline Completes the CCSAS CSE Implementation Planner Serves as a representative to LCSA Group Meetings (comprised of LCSAs grouped by implementation date/region)
Counterpart to the designated LCSA Change Management Liaison Supports the development of county outreach and orientation approaches Facilitates the completion of the LCSA's business process analysis Supports the conduct of user orientation sessions	Change Management	Customizes communications approach and materials (outreach and orientation) Coordinates business process analysis activities; decides how baseline process flows will be implemented in county Plans and conducts user orientation sessions to inform users of the new system and orient them to changes prior to training (coordinates orientation meeting, newsletters, etc.)

Alliance Team Member Responsibilities	Site Implementation Team Role	LCSA Liaison Responsibilities
Counterpart to the designated LCSA Data Conversion Liaison; supports LCSA Data Conversion Liaison in their tasks Participates in conducting LCSA-specific conversion activities, including data mapping, data cleansing, data acceptance testing, and conversion cutover	Data Conversion	Works with the alliance Data Conversion team member to plan and coordinate conversion activities that require LCSA staff participation Identifies and coordinates the assignment of staff required to support data mapping, data cleansing, data acceptance testing, and conversion cutover support
Counterpart to the designated LCSA HW/SW Installation Liaison; supports LCSA HW/SW Install Liaison in their tasks Manages the installation of cabling, hardware, and software in the LCSA Participates in support of site technical staff	Hardware/Software Installation	Acts as primary point of contact to alliance HW/SW team members Allows access to facility Validates that HW/SW install is finalized and criteria are met
Counterpart to the designated LCSA User Training Liaison; supports LCSA User Training Liaison in their tasks Determines training site and schedule for LCSA users Delivers instructor-led classroom training and facilitates coach-led e-learning sessions for LCSA users Delivers training attendance reports and evaluation	User Training	Identifies users to be trained and the type of training they will receive Schedules and communicates training schedule to users to facilitate attendance User reference: assists in development of LCSA/site specific user reference materials, as needed
Counterpart to the designated Local Interface Liaison; supports LCSA Local Interface Liaison in their tasks Works with county IV-A agencies Verifies LCSA's interfaces are working properly prior to cutover	Local Interfaces	Identifies interfaces Works with the project's counterpart to identify the data to map to the standard interface formats Verifies local interfaces' connectivity

Alliance Team Member Responsibilities	Site Implementation Team Role	LCSA Liaison Responsibilities
Counterpart to the designated LCSA On-Site Support Liaison and "superusers"; supports LCSA On-Site Support Liaison in their tasks Coordinates LCSA set-up tasks and configurations (user security profiles, drop-down values, printers) Prepares plan for On-Site Support Facilitates the development of an LCSA-specific issue resolution paths Orients LCSA "superusers" Leads post-cutover status meetings to assess system transition	On-Site Support	Defines CCSAS CSE user population and identifies and assigns CCSAS CSE security profiles to users Develops approach for LCSA/site post-cutover issue resolution paths (both during period of on-site support and after), including transition strategy for after the project on-site support team leaves Identifies staff to serve as LCSA "superuser" team, as appropriate Defines LCSA-specific reference data Participates in on-site user support during transition to CCSAS CSE and leads user support after CCSAS CSE team leaves

Figure 3C.13.2.3-11 Responsibilities of the Site Implementation Team - *Staff from the alliance and the State are assigned to represent each of the seven disciplines. These staff work together to support the implementation of each LCSA.*

Implementation Steps

Major planning events in each LCSA are organized along distinct implementation steps. The four implementation steps are:

- Start-up
- Readiness
- Pre-cutover
- Post-cutover

Figure 3C.13.2.3-12 depicts the key planning events by each of the four steps of implementation.

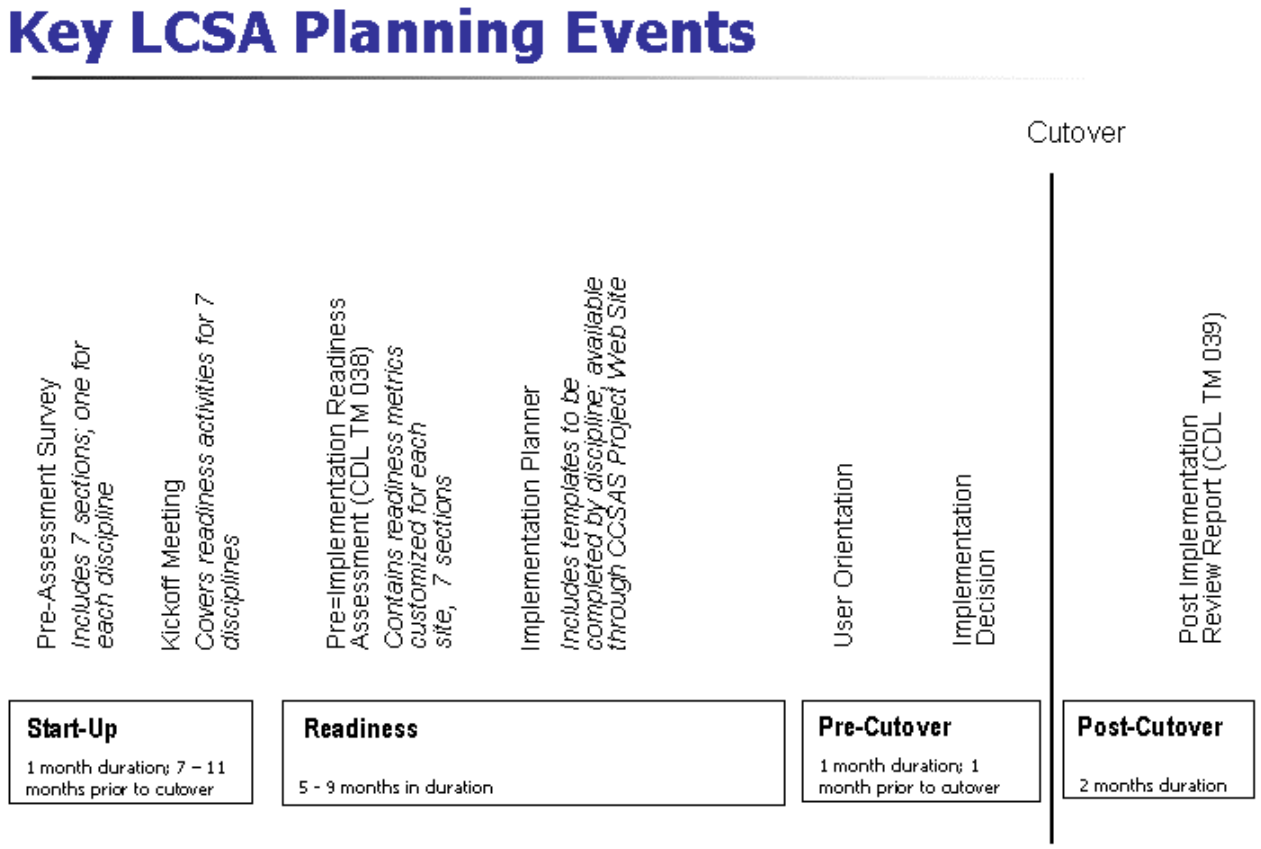


Figure 3C.13.2.3-12 LCSA Key Planning Events - *Events in the LCSAs are coordinated among the seven disciplines to reduce the number of "touchpoints."*

The duration of time it takes to implement an LCSA is dependent upon the size of the LCSA as measured by the number of users. This metric is used to account for the relative complexity of preparing larger county organizations for transition. While the implementation planning period takes place over a number of months, it is not anticipated that CCSAS project staff, alliance staff, or LCSA staff will be working full time on implementing the new system during this preparation period.

In order to limit disruption, we will hold a series of key events in each LCSA in which representatives from the six disciplines participate as presenters. One of the roles of the Site Implementation Manager is to coordinate these key events. In addition to the key events, there will be specific times when staff from one discipline will be in the LCSA working on discipline-specific readiness activities.

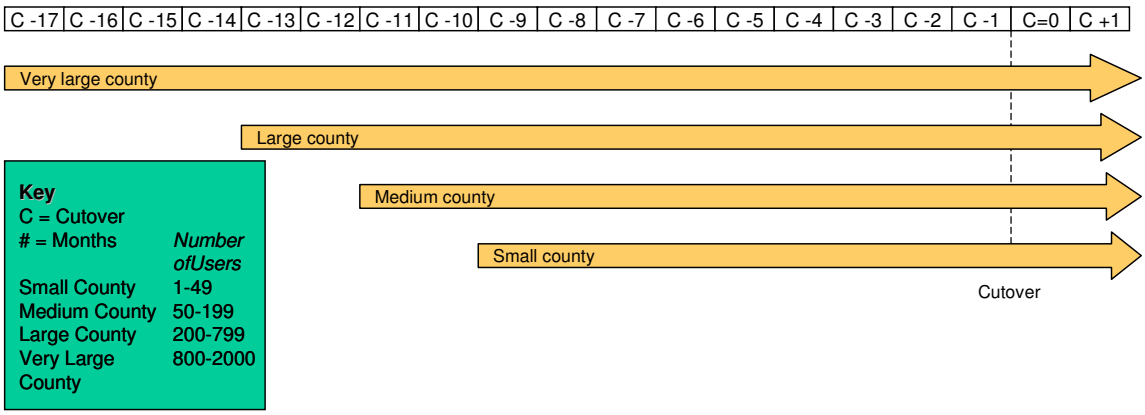


Figure 3C.13.2.3-13 Timing for LCSA Planning - *Planning activities take between seven and eleven months, depending on the size of the county.*

Start-Up Step

In the Start-Up Step, the alliance will complete Pre-Assessment Surveys early in the project. CCSAS CSE work is initiated in the LCSA with a kick-off meeting about 7 – 11 months prior to cutover, depending on the size of the LCSA. The key events in the Start-Up Step are:

- This is a survey conducted early in the project that gathers key information from LCSAs as a baseline for beginning CCSAS CSE work. It is composed of several subsections organized by discipline, each subsection capturing important baseline information about the county. For example, key planning questions are: "Who will act as the key points of contact for your LCSA, by discipline?"; "How many users are in your LCSA, by role?"; and "How many PCs are needed for your LCSA, by site?"
- A Kickoff Meeting is held involving the staff that will be working on implementation activities for that LCSA. The focus of the kick-off meeting is to introduce and discuss the implementation processes, activities and tools that are used in preparing and the LCSA for transition.
- For the month after kickoff, work in the LCSA focuses on customizing the Pre-Implementation Readiness Assessment (CDL TM 038).

Readiness Step

The Readiness Step proceeds over the next 5 - 9 months (depending on the LCSA size). During this period, the State, LCSA Site Coordinator and the Site Implementation Manager oversee change management efforts, customize the Readiness Metrics for the LCSA, and prepare for cutover. Initially, LCSA involvement may include one CCSAS CSE meeting per week at the LCSA. As the site moves closer toward the cutover date, alliance staff will meet together more frequently with the State and LCSA staff to discuss and resolve issues related to achieving readiness as measured by the Readiness Metrics.

- The Pre-Implementation Readiness Assessment (CDL TM 038) is an assessment of each LCSA performed 6 months prior to implementation. The Pre-Implementation Readiness Assessment, resulting in a document similar to an implementation plan, is monitored throughout the implementation preparation effort at three months, two months, and one month prior to implementation. The Readiness Assessment process will lead to an implementation readiness review and an implementation/no implementation decision. The Readiness Metrics are reviewed and customized in the beginning of the planning effort to help participants and LCSA personnel understand the prerequisites for transition.
- The Implementation Planner serves as a roadmap for use by LCSA, and combined alliance/CCSAS project staff while preparing to deploy CCSAS CSE in each site. Specifically, the Implementation Planner provides a series of templates available through the CCSAS CSE project web site to help LCSA and combined alliance/CCSAS project staff perform readiness activities for each of the seven disciplines (e.g., conversion, training, etc.) Eventually, the completion of the templates in the Implementation Planner leads to the satisfaction of the Readiness Metrics.

Pre-Cutover Step

In the month prior to cutover, the focus of activities moves toward assessing the completion of the LCSAs Readiness Metrics and the determination to move forward with implementation.

- The SCP states that users should be “oriented” prior to training. Therefore, leadership will conduct an event to inform users about the CCSAS CSE system, implementation process, and status..
- Ten days prior to cutover, the CCSAS project representative, the LCSA Director and LCSA Site Coordinator, and the Site Implementation Manager will conduct the implementation readiness review to assess whether the Readiness Metrics are finalized and to decide whether to move forward with scheduled implementation activities.

Post-Cutover Step

In the Post-Cutover Step, staff working on CCSAS CSE in the LCSA supports the transition to production. The Post-Cutover Step lasts about two months.

- ***Post-Implementation Review Report (CDL TM 039):*** This report, due 1 month after each implementation, documents the implementation methodology and issues/problems encountered. It contains lessons learned, successes and failures, and timeframes/slippages, if any. It also contains production issues that arise during implementation due to business transition (including work-arounds). The results of these reports are incorporated into our future implementation approach for subsequent LCSA transitions in order to leverage any lessons learned.

Figure 3C.13.2.3-14 [RESERVE]

Regional and Group Meetings

In addition to the work we conduct at each LCSA to prepare them for implementation, we also propose holding regular meetings to keep LCSA officials informed and involved in the CCSAS CSE project. These meetings are:

- CCSAS CSE Directors' Meetings
- LCSA Group Meetings

CCSAS CSE Directors' Meetings

The Directors' Meetings present an opportunity for the DCSS officials and Regional Managers, CCSAS project, and the alliance to keep LCSA Directors informed of project developments. Under the leadership of the CCSAS project and DCSS, the Directors' Meetings allow LCSA Directors to become more engaged in upcoming project initiatives. This is particularly important in the beginning stages of the project, where the LCSA Directors are the primary representatives from the counties involved in the CCSAS CSE project. The LCSA Director is key in the communications process. The responsibilities of the Director are to:

> Recognizing the importance of meaningful sponsorship from LCSA Directors, the alliance conducts semi-annual CCSAS CSE Directors' Meetings to keep the Directors informed of project progress and developments.

- Designate CCSAS CSE as an LCSA priority
- Define and communicate LCSA vision, goals, and objectives for implementing and using CCSAS CSE
- Identify LCSA staff to be involved in CCSAS CSE preparation and transition; assigns roles and responsibilities
- Participate in CCSAS CSE Directors' Meetings

LCSA Group Meetings

To keep the LCSA Site Coordinators for CCSAS CSE engaged and to create synergy among counties, we propose supporting the CCSAS project and DCSS in holding LCSA Group Meetings. These LCSA Groups will meet regularly with state and project staff to remain informed of project and implementation status, validate the implementation approach and materials, and support each other in their transition to CCSAS CSE and CASES.

> To support the planning process, the implementation team conducts LCSA Group Meetings with LCSA Site Coordinators to communicate relevant project developments and gain their insight and assistance on implementation-related project activities.

The LCSA Group Meetings will be a forum to discuss ongoing monitoring and refinement of the implementation approach. The alliance has found this level of LCSA involvement to be an invaluable way to prepare for and support statewide system implementations. We have worked with a similar group of county representatives on the eICMS project in Ohio. We found these group meetings to be an

excellent forum for statewide implementation discussion and sharing of experiences.

The first LCSA Group formed in each phase of the project is the group of pilot counties. Planning for the pilot may require increased effort and involve issues that are specific to piloting a new system. As we move on to subsequent implementations beyond the group of pilot counties, additional LCSA Groups will be formed. To support communication and coordination with new LCSAs as their implementation date nears, we will work with the CCSAS project and the existing LCSA Groups to define a process for involvement. Staff from LCSAs in earlier implementations may act as advisors to staff from LCSAs with upcoming implementation dates.

Meeting	Participants	Objective	Timing
CCSAS CSE Directors' Meetings	LCSA Directors from 58 LCSAs; Directors meet by region.	To allow DCSS leadership and Regional Managers and the CCSAS project the opportunity to keep LCSA Directors informed of project status, system design, and proposed implementation approach.	Approximately two times per year, per region Where possible, Directors' Meetings will take advantage of existing statewide or regional meetings of LCSA Directors, however, Directors' Meetings focused on CCSAS CSE can offer more personal contact and targeted subject matter.
LCSA Group Meetings	LCSA Coordinators from 58 counties split into 6 groups based on implementation date and geographic location (i.e., the first LCSA group will likely be comprised of the pilot LCSAs).	To allow CCSAS project leadership the opportunity to keep LCSA Coordinators informed of project status, system design, and proposed implementation approach; allows an opportunity for LCSA Coordinators to validate our proposed implementation approach and to learn from other LCSA's implementation experiences.	Approximately two times per year, per group, although LCSA groups may meet more frequently in the months prior to and after their implementation dates and may meet less frequently during other points in the project.

Figure 3C.13.2.3-15 Group Meetings Involving LCSA Officials – *This table depicts the ways in which we propose to meet with groups of LCSAs to keep them informed and involved during the course of the project.*

3C.13.2.3.3 [RESERVE]

3C.13.2.4 Transitioning Users and Functions

The alliance is proposing to transition California's child support system users to the CCSAS CSE statewide system in two phases. This phasing strategy allows for a controlled implementation of functionality and transition, and cost savings from early retirement of child support consortia systems.

During Phase I, 14 counties will transition from the KIDZ, and STAR/KIDS consortia to the CASES consortium system. Fifty-eight counties from the CASES and ARS consortia will interface with our proposed Statewide Services component, Version 1 of the CSE system. This Version 1 configuration will provide DCSS with the ability to improve program performance early in the project's life cycle.

Version 2 builds on the functionality implemented during Version 1 and will replace the local ARS and CASES implementations with our best of breed system. At the end of Phase II the CCSAS CSE will be a single statewide system.

The Business Process Transition Plan (TM 056) will document the high-level transition strategy to the CSE system.

Transitioning Users

The CCSAS CSE core user communities affected by our proposed phased implementation include the following groups:

- *Department of Child Support Services (DCSS)*. This is the State level organization and staff that provides child support service delivery planning, program management, and oversight.
- *Local Child Support Agency (LCSA).* These are the case, financial, and support professionals who perform child support program activities at the counties.
- *Department of Justice: California Central Registry (CCR)*. This is the State Department of Justice office and staff that operates as a clearinghouse for incoming interstate child support cases.
- *Department of Justice: California Parent Locator System (CPLS).* This is the group that processes locate requests from various out of state jurisdictions and the LCSAs.

In addition to these core user groups, there are numerous supporting user groups, as well as the end recipients of child support services.

Transitioning Functions

The functions to be transitioned discussed in the following sections can be placed into one of six distinct categories:

- *State Level Services.* These services are centrally operated, providing data and processing services to the Child Support Program as a whole.

- *LCSA Level Services.* These are services that support business processes traditionally performed at the local LCSA level. Examples of LCSA services include case initiation, establishment, and enforcement functions.
- *Local Agency Key Interfaces.* These are interfaces with local entities that share data with the LCSAs to support welfare and county court processes.
- *State Agency Key Interfaces.* These are interfaces with other state and local entities which share data with the core Child Support Program agencies to support locate and enforcement processes.
- *Federal Agency Key Interfaces.* These are interfaces with Federal entities which share data with California's Child Support Program to support locate and enforcement processes.
- *Private Sector Entity Key Interfaces.* These are interfaces with non-governmental entities such as employers and credit reporting agencies.

"Snapshots" are used throughout this discussion to illustrate the child support functions, users, and interfaces at key points during the project. These depictions are not milestones. Instead, they convey the proposed transition of functions and users at certain points within the overall project lifecycle.

3C.13.2.4.1 Transitioning Users and Functions

The primary objectives of Phase I are to:

- Implement Version 1 of CCSAS CSE, including the State Case Registry (SCR), extended locate services, federal reporting, SDU interface, and CSENet interface functions.
- Transition to a two-consortia environment, by converting the KIDZ and STAR/KIDS consortia LCSA to the CASES consortia system.
- Enable the State of California to begin realizing benefits and increase program performance early in the project lifecycle.

The primary objectives of Phase II are to:

- Implement the federally certifiable CCSAS CSE Version 2 statewide child support enforcement system in LCSAs and DCSS.
- Improve the quality, consistency and timeliness of the delivery of services to recipients of CSE program services.
- Reduce the complexity, cost, and time required to operate, manage and maintain the CSE system, interfaces and technical infrastructure.
- Enable the State of California to realize benefits in the CSE program through increased collections, improved customer service, reduced cost of maintenance and operation, and increased worker effectiveness. Construct the LCSA-level functionality of the best of breed Version 2 solution, which when integrated with Version 1, provides the State with CCSAS CSE, a single statewide system.

The following sections describe the strategy for transitioning users and functions to CSE Versions 1 and 2.

Figure 3C.13.2.4-1 [RESERVE]

Figure 3C.13.2.4-2 [RESERVE]

Figure 3C.13.2.4-3 [RESERVE]

Figure 3C.13.2.4-4 [RESERVE]

Snapshot 1

"Snapshot 1" depicts the child support environment after transition of the 14 KIDZ and STAR/KIDS LCSAs to the CASES consortia. It also shows the deployment of the CCSAS CSE Version 1 functionality for the State Case Registry (SCR), locate, federal reporting, SDU interface and CSENet interface functions.

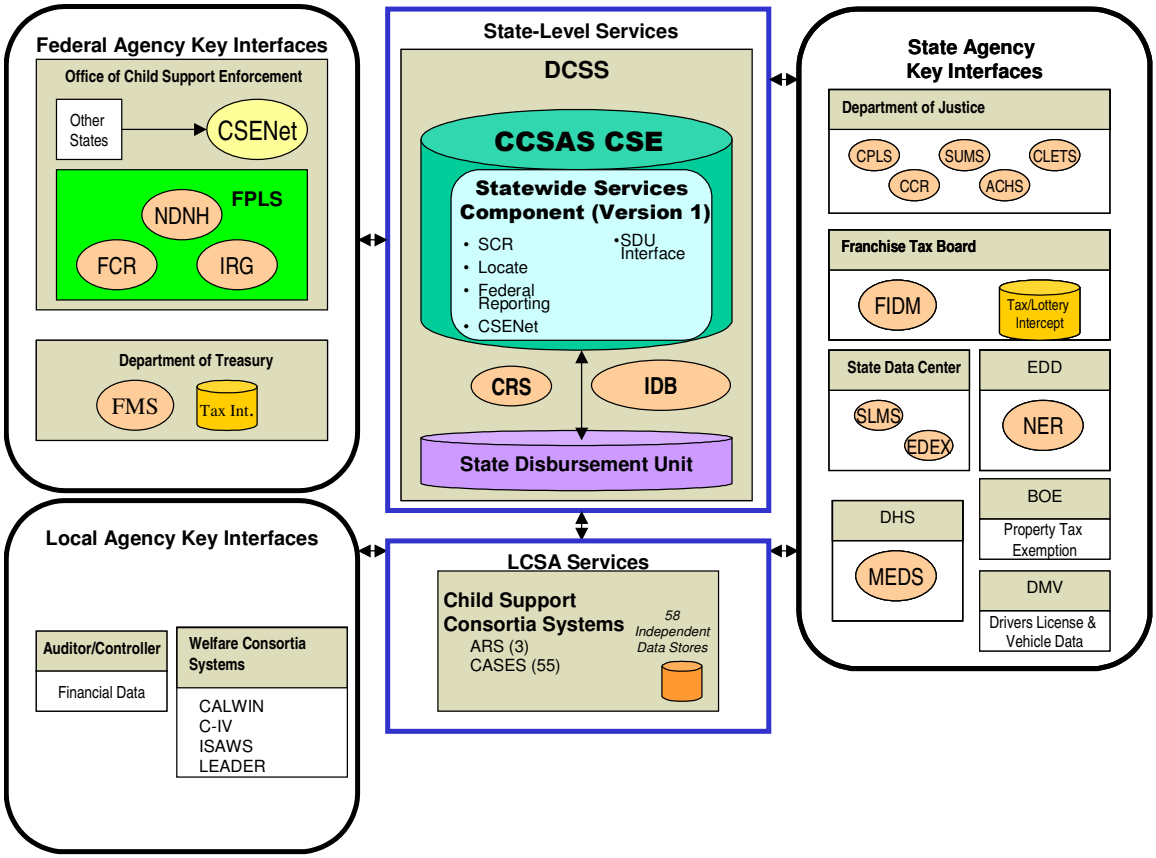


Figure 3C.13.2.4-5 Snapshot 1 - *High-level depiction of functions and users after Phase I rollout.*

The following tables in Figure 3C.13.2.4-6 introduce CCSAS CSE as a function within the Department of Child Support Services.

State Level Services	
Department of Child Support Services	
DCSS is designated as the single state entity responsible for operating the Child Support Enforcement Program. DCSS, through their agent FTB, is responsible for the operations and maintenance of the CCSAS CSE Statewide Services Component deployed at the end of Phase I.	
Version 1 of CSE System	The alliance will develop Version 1 of the CCSAS CSE system, representing the first release of CCSAS CSE application functionality. Version 1 will be built within our proposed best of breed technical architecture, providing the state-level functionality of our best of breed solution. The alliance will implement a new CCSAS CSE Data Center to house operations for Version 1. Version 1 will incorporate improved Locate processing, Federal reporting capabilities, the State Case Registry (SCR) , the State Disbursement Unit (SDU) interface,

State Level Services	
	and CSENet functionality.
SCR	The SCR function of the CCSAS CSE Statewide Services component transmits data to and receives FCR information. With the completion of Version 1 for the State Case Registry (SCR), the alliance will begin populating the SCR database. Population of the SCR database enables Version 1 for locate, reporting, interstate and the SDU interface to be implemented during Phase I of the project. The SCR is the first step in what will ultimately become the fully integrated single statewide database, containing IV-D information from LCSAs and Non IV-D orders converted from the NICE database. The integration provided by the SCR delivers uniformity of structure to data elements stored in the SCR. DCSS and the LCSAs will, for the first time, have on-line access to a single location in which individual, case, employment and financial information is stored. Further, the SCR provides this information uniformly and subjects the information to standardized, consistently applied edits and validations.
Locate	The first of the statewide locate capabilities will be constructed during Phase I. This functionality will leverage the SCR to improve and consolidate locate requests and results statewide. The information collected and stored in the SCR will provide an indication as to the availability of current identity, locate, employment and asset information for individuals. For example, address information in the SCR will indicate if the information is current, and whether the information has been verified. The Statewide Services locate component will monitor information and will initiate locate requests to obtain and verify identity, location, employment and asset information. The locate component will monitor requests and will receive, process, consolidate and distribute responses received. The locate component will leverage the information sources accessed through California Parent Locater Service (CPLS). For locate sources already in use in the current environment, the statewide services locate component will route locate requests through the current CPLS interface. The locate component will receive responses from CPLS, consolidate and sequence the responses, and forward them to LCSAs that have a case(s) involving the individual to whom the locate request and response(s) are related.
Federal Reporting	CCSAS CSE becomes the federal reporting agent between LCSAs and DCSS. CCSAS CSE will provide a web-based interface for the LCSAs to use to submit information for the OCSE-34A and OCSE-157 reports. The reporting interface will include the functionality necessary to integrate and consolidate information received into the required statewide view of information. The reporting component will also include the functionality necessary to process the federal audit extract file, and compare and validate results received from the LCSAs. The federal reporting features provide a means by which information submitted by the LCSAs are stored and consolidated for the purpose of creating reports to monitor trends.
CSENet	The first of two CSENet components will be included in Version 1 of the CSE system. This component includes the functionality to support two CSENet transactions: the Quick Locate transaction and the Case Status Inquiry (CSI) transaction. Other inbound CSENet transations are received by the CSE system and forwarded to CASES and ARS. Other outbound CSENet transactions are received by the CSE system from CASES and ARS and are transmitted to CSENet.

State Level Services	
State Disbursement Unit (SDU) Interface	DCSS plans to separately procure a vendor to perform the collection and disbursement functions typical of an SDU provider. Phase I of the project includes the implementation of two interfaces related to the SDU. The first interface provides for the transfer of collection information between the SDU solution and the statewide system, and the transfer of disbursement instructions between the statewide system and the SDU for Non-IV-D orders. The second interface provides for the transfer of allocated collection information between the statewide system and the LCSAs.

LCSA Services	
Child Support Consortia Systems	To provide true statewide functionality, information must be shared between the CCSAS CSE Statewide Services component and the local ARS and CASES systems. A new interface between the consortia systems and the CCSAS CSE system will supply information for the State Case Registry from the consortia systems to the CCSAS database.

Figure 3C.13.2.4-6 CCSAS CSE as a Function within the Department of Child Support Services – *These tables describe the interaction of CCSAS CSE with State Level Services and LCSA Services.*

Figure 3C.13.2.4-7 [RESERVE]

Figure 3C.13.2.4-8 [RESERVE]

Figure 3C.13.2.4-9 [RESERVE]

Figure 3C.13.2.4-10 [RESERVE]

Figure 3C.13.2.4-11 [RESERVE]

Figure 3C.13.2.4-12 [RESERVE

Figure 3C.13.2.4-13 [RESERVE]

Figure 3C.13.2.4-14 [RESERVE]

Snapshot 2

"Snapshot 2" shows the full implementation of the CCSAS CSE system after the implementation of Version 2. This is the result of the implementation of both Version 1 I and Version 2 and meets the objectives outlined for both phases of the project. The alliance will transition 58 LCSAs to the CCSAS CSE system during Phase II. The alliance's proposed sequence and schedule for transition is detailed in Section 3C.13.2.4.2 LCSA Rollout Approach.

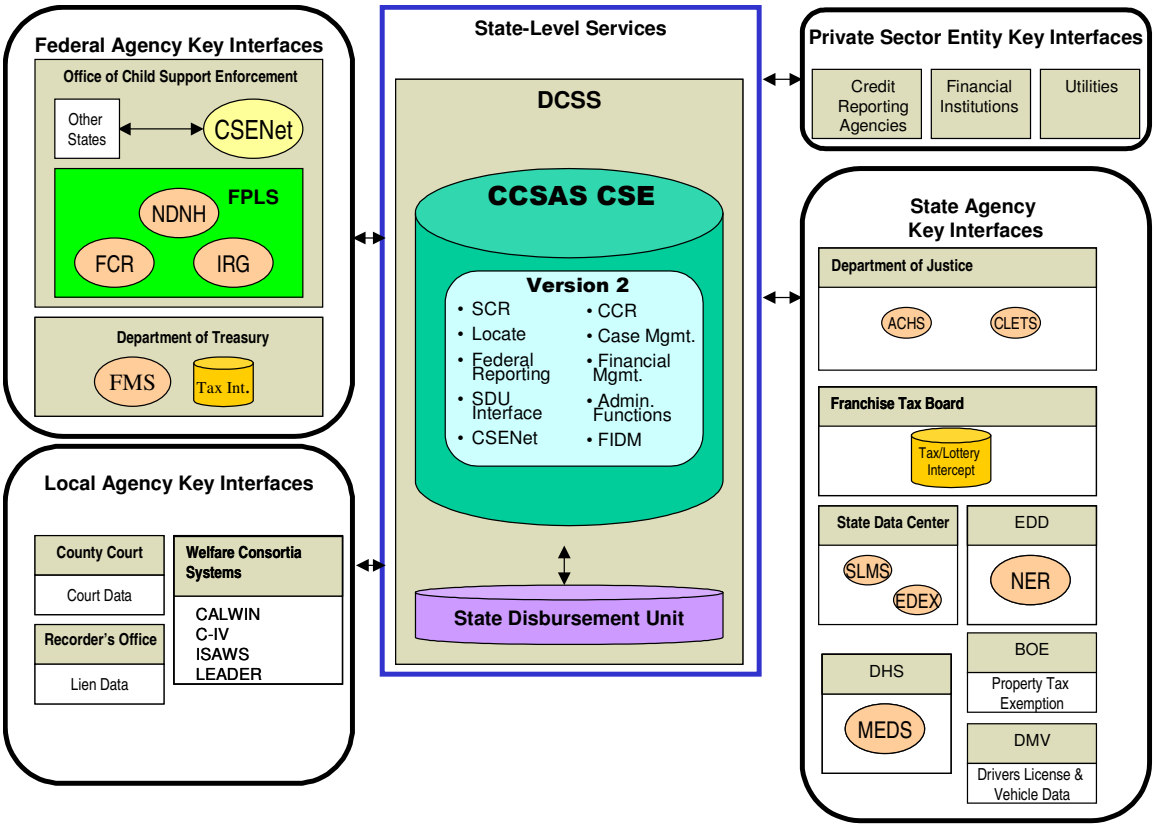


Figure 3C.13.2.4-15 Snapshot 2 –*High-level depiction of users and functions after full implementation of CCSAS CSE.*

The following tables in Figure 3C.13.2.4-16 describe the transition of functions and users, along with the establishment of new interfaces upon full implementation of the CCSAS CSE Version 2 statewide system.

State Level Services	
Department of Child Support Services	
DCSS is designated as the single state entity responsible for operating the Child Support Enforcement Program. DCSS, through their agent FTB, is responsible for the operations and maintenance of the CCSAS CSE deployed as Version 2 at the end of Phase II.	
CCR	The California Central Registry functionality will be transitioned to CCSAS CSE in parallel with the transition of LCSAs to CCSAS CSE.
Case Management Financial Management Administrative Functions	These functions will be performed by Version 2 of the CCSAS CSE application.
FIDM	Version 2 will implement the system functionality and supporting processing for the Financial Institution Data Match (FIDM) program. This functionality will be replaced by functionality contained in CCSAS CSE.

Key Interfaces: Local Agencies	
County Court County Recorder's Office Welfare Consortia Systems	The local entities exchange information with CCSAS CSE.

Key Interfaces: State Agencies	
Department of Justice	CCASS CSE interfaces with DOJ for criminal history information (CLETS).. The functions provided by the California Parent Locator System (CPLS) are integrated into the CCSAS CSE system during Phase II implementation. Additionally, the functions provided by the California Case Registry (CCR) are integrated into the CCSAS CSE system and are transitioned from DOJ to DCSS.
Franchise Tax Board	The Franchise Tax Board (FTB) is the official taxing authority for the State of California. FTB Intercepts non-custodial parents' state income tax refunds and Lottery winnings to satisfy support arrearages. With the implementation of the statewide system, the state tax intercepts are submitted directly to the FTB by the CSE system. Delinquent cases are referred to FTB through a batch file submission.
Employment Development Department (EDD)	The Employment Development Department (EDD) administers the Job Service, Unemployment Insurance, Disability Insurance, Workforce Investment Act, and Welfare-to-Work programs. California employers are required to report new and returning employees within thirty (30) days of being hired, quarterly wage information, unemployment insurance and disability insurance intercepts. CCSAS CSE interfaces directly with EDD for these data.
Board of Equalization (BOE)	The BOE sends property tax data directly to CCSAS CSE.

Department of Motor Vehicles (DMV)	The DMV sends drivers license and vehicle registration data along with address information directly to CCSAS CSE.

Figure 3C.13.2.4-16 Transition of Functions and Users - *These tables describe transition, along with the establishment of new interfaces upon full implementation of CCSAS CSE.*

Figure 3C.13.2.4-17 [RESERVE]

Figure 3C.13.2.4-18 [RESERVE]

3C.13.2.4.2 LCSA Rollout Approach

The following considerations and criteria aim to reduce implementation risk through the effective utilization of staff, careful planning to reduce dependencies, and the evaluation of certain "readiness" criteria to increase LCSA buy-in and improve the project's probability of success:

- Organize based on consortia
- Accelerate reduction of maintenance & operations (M&O) costs
- Implement consortia leads last
- Reduce disruption and adverse impact of change
- Limit dependencies on old welfare systems
- Load balance the caseload being transitioned
- Allocate staff efficiently based on geography
- Build in time for evaluation and issue resolution

Organize Based on Consortia

To increase the effectiveness of the implementation teams, the alliance proposes to organize the rollout in multiple tracks based on consortia. This allows the disciplines (e.g., conversion and change management) to become experts in the source system such that they are able to achieve increases in efficiency through specialization. In this way, the implementation teams can take advantage of their specialized knowledge of the consortia systems.

Accelerate Reduction of Maintenance & Operations (M&O) Costs

One of the key objectives of the alliance implementation strategy is to eliminate the maintenance and operation costs associated with the consortia systems. Early retirement of consortia systems has potential for program savings, effectively reducing the cost of the CCSAS CSE project and implementation of a single statewide system.

Implement Consortia Leads Last

During the overall implementation of the CCSAS CSE project, six separate consortia systems, each supporting two or more LCSAs must be retired from operation. The alliance rollout approach recognizes the need to continue operations and perform ongoing system maintenance until the last county using a specific consortia system has transitioned. As such, the lead county for each consortium is scheduled last within the consortia track to provide the necessary maintenance and operation services. Once the lead county has transitioned off the consortia system, the consortia and its associated M&O costs are retired.

Reduce Disruption and Adverse Impact of Change

To reduce disruption and increase LCSA buy-in, the most recently transitioned LCSA offices are placed as near the end of the rollout schedule as possible. The objective of this "last-in last-out" scheduling criterion is to create as much time between system

transitions as possible in an effort to reduce the adverse impact of change on users and program performance.

Limit Dependencies on Old Welfare Systems

Several LCSAs are scheduled to transition to a new welfare consortia system over the next two to three years. The transition of these LCSAs to a new welfare system requires the development and implementation of new interfaces between the welfare and IV-D systems.

To limit dependencies between the child support enforcement system and the new welfare consortia systems, the alliance rollout strategy places implementation of LCSA offices in those counties after that of the new welfare system. This reduces the possibility that the CCSAS CSE project must implement the welfare interface for a particular county twice. As of this writing, the county welfare transitions are scheduled to occur before rollout of the child support enforcement system during Phase I. Some, however, may be close with respect to timing. To further reduce the risk of implementation, the rollout schedule allows as much lead time as possible for the welfare systems to stabilize before implementing the child support enforcement system in those LCSA sites.

Load Balance the Caseload Being Transitioned

To balance the workload for implementation staff, the rollout schedule strives to level the effort required for rollout by controlling the number of users and cases that are transitioned to a new system each month. To further mitigate risk, the rollout schedule is constructed such that workload increases or ramps up during the first months of implementation. This results in placing smaller LCSA offices at the head of the schedule and increasing the size and number of offices that are transitioned in subsequent months until a predetermined steady state is achieved.

Conversely, placing large LCSAs at the head of the schedule increases the risk of implementation failure as these, by virtue of their size; represent the most complex and difficult offices to implement. Attempting to implement too many LCSA offices during the first months will increase implementation risk.

Allocate Staff Efficiently Based on Geography

To contain travel and the effects it may have on implementation staff, the rollout schedule groups LCSAs within geographic proximity into similar tracks. This approach additionally serves to: (1) keep implementation teams within close geographic proximity to recently transitioned offices should they require additional assistance; (2) help implementation teams gain a better understanding of the LCSAs within an area given they may have similar geography, demographics, and issues; and (3) physically groups activities such that staff are more readily shared amongst LCSA implementations.

Build In Time For Evaluation And Issue Resolution

To further mitigate the risk of implementation failure, the schedule includes time for evaluation, incorporation of lessons learned, and issue resolution. Our experience with large-scale system implementations, similar to CCSAS CSE, dictates the inclusion of "rest periods" – months in which no new offices are scheduled for transition. In addition to using this time for evaluation, this time can serve as a "catch up" period for other project tasks if necessary.

3C.13.2.4.2.1 Phase I Rollout

The proposed Phase I rollout schedule is comprised of two segments: a two-month pilot implementation followed by a 9-month, non-pilot rollout period.

Phase I Pilot Strategy and Selection

The primary objective of the Phase I Pilot is to assess the effectiveness of our proposed implementation methodology. Normally, pilot periods are also used to evaluate new systems in a production environment; however, with CASES, this need is mitigated given the maturity of the application. This allows the team to focus on incorporating lessons learned into the implementation methodology without having to deal with the instabilities that are usually present with a new system.

In developing the Phase I pilot schedule and selecting the pilot LCSAs shown in the following section, the alliance took several factors into consideration:

- **Consortium** – The LCSAs selected for the Phase I CASES pilot is split among the two transitioning consortia allowing the implementation team to apply lessons learned from one county representative to the next.
- **Readiness** – The pilot includes LCSAs with the best-demonstrated performance (relative to federal indicators) under the assumption that these LCSAs are the most ready (i.e. not likely to have caseload backlogs) to adapt to a new system.
- **Eliminate Consortia** – In keeping with the objective to accelerate the reduction of M&O costs, the pilot targets LCSAs from consortia that are scheduled for elimination early in the project. By eliminating consortia, the State: a) reduces M&O costs given that the economies of scale decline as the number of LCSAs using a particular consortium system decreases; b) requires IV&V oversight for a smaller number of consortia; c) reduces system enhancement/maintenance costs in the event the new State and/or Federal CSE requirements are issued prior to statewide rollout of the new system.
- **Last in Last Out** – In keeping with the objective to reduce disruption and the impact of change, LCSAs that recently joined their selected safe haven consortium are not considered in the Phase I pilot.
- **Exclude Large LCSAs** – Large LCSAs, given their large caseloads and user populations, are not good candidates for pilot consideration.
- **Exclude Consortia Lead** – LCSAs that hold contracts with the consortium's M&O vendor(s) are not implemented before the other LCSA in the consortium as this would violate the previously discussed "implement consortia leads last" criteria.

- **Difficult Accessibility** – LCSAs that are difficult to reach were excluded given that: a) quick deployment of resources may be necessary in a pilot implementation; and b) other county and State personnel will need easy access in the event they choose to visit the pilot site.

Phase I Pilot Schedule

As shown in Figure 3C.13.2.4-19 below, the two-month Phase I pilot implementation transitions two LCSAs in month two. Month three is reserved for the evaluation of lessons learned and the incorporation of those lessons into the implementation methodology, tools, and procedures.

Cutover Month	LCSA	Region	Consortium	Approximate # of Users Converted
1	No LCSAs scheduled			
2	Placer	3	KIDZ	90
	Santa Barbara	5	STAR/KIDS	125
3			Evaluation	0

Figure 3C.13.2.4-19 Phase I Pilot Rollout Schedule - *Proposed Phase I Pilot Implementation Sequence and Schedule by LCSA.*

Phase I Non-pilot Schedule

Once the pilot implementation is concluded and lessons learned are incorporated into the implementation methodology, tools, and procedures, the remaining 12 LCSAs, as shown in Figure 3C.12.2.4-20 below, are transitioned to CASES over a nine-month period. The proposed rollout schedule uses the scheduling approach described in Section 3C.13.2.4.2. The application of our approach may, however, result in many possible outcomes. The alliance intends to work closely with DCSS and CCSAS project management to determine the actual, desired rollout sequence given the many factors, such as regionalization and minimizing county disruption, that can influence the outcome.

Cutover Month	LCSA	Region	Consortium	Approximate # of Users Converted
1	No LCSAs scheduled			
2	Yolo	1	KIDZ	80
	Yuba	1	KIDZ	30
3	Shasta	1	KIDZ	115
4	San Bernardino	5	STAR/KIDS	460
5	Butte	1	KIDZ	140
	Mendocino	1	KIDZ	40
6	San Joaquin	4	KIDZ	200

Cutover Month	LCSA	Region	Consortium	Approximate # of Users Converted
	Imperial	5	KIDZ	40
7	Ventura	5	KIDZ	265
	Fresno	4	STAR/KIDS	310
8	No LCSAs scheduled			0
9	Riverside	5	STAR/KIDS	475
	Kern	5	KIDZ	250

Figure 3C.13.2.4-20 Phase I Rollout Schedule - *Proposed Phase I Implementation Sequence and Schedule by LCSA.*

The table above represents the rollout sequence, reflecting the month of cutover. It does not reflect the lead-time necessary to prepare for implementation.

3C.13.2.4.2.2 Phase II Rollout

Phase II of the proposed implementation schedule rollout is the alliance's Version 2 system to 58 LCSAs over a 24 - 30 month period. The Phase II schedule itself is comprised of two segments: a four-month pilot implementation followed by a 20-month statewide rollout period.

Phase II Pilot Strategy and Selection

The objective of the Phase II pilot is to evaluate the full range of system functionality for the best of breed system in a production environment prior to beginning statewide rollout. In addition, we will validate our proposed implementation methodology when applied to counties transitioning to the best of breed system.

Pilot implementation includes the use of the Statewide Disbursement Unit (SDU) to validate interfaces. Based on the proposed implementation schedule and phased approach, the SDU will be fully tested, accepted, and operational on or before the start of pilot implementation.

In developing the Phase II pilot strategy and corresponding implementation schedule; the alliance considered several factors:

- **Business Model** – The new system must be shown to be flexible enough to support LCSAs with differing business models ranging from specialized (typically larger LCSAs) to generalized (typically smaller LCSAs) operations. Therefore, the pilot includes LCSAs with varying user population sizes.
- **Readiness** – The pilot includes LCSAs with the best-demonstrated performance (relative to federal indicators) under the assumption that these LCSAs are the most ready (i.e. not likely to have caseload backlogs) to adapt to a new system.
- **Last in Last Out** – In keeping with the objective to reduce disruption and the adverse impact of change, LCSAs that recently joined their selected safe haven consortium are be considered as a candidate for Phase II pilot.

- **Exclude Large LCSAs** – Large LCSAs, given their large caseloads and user populations, are not good candidates for pilot consideration.
- **Exclude Consortia Leads** – LCSAs that hold contracts with the consortium's M&O vendor(s) are not implemented before the other LCSA in the consortium as this would violate the previously "implement consortia leads last" criteria.
- **Difficult Accessibility** – LCSAs that are difficult to reach were excluded given that: a) quick deployment of resources may be necessary in a pilot implementation; and b) other county and State personnel will need easy access in the event they choose to visit the pilot site.

Phase II Pilot Schedule

As shown in Figure 3C.13.2.4-21, the four-month pilot implementation transitions two LCSA in month one with two additional LCSAs implemented in month two. Months three and four of the pilot implementation are reserved for evaluation of the new CSE system and issue resolution. A joint go/no-go decision is made in cooperation with CCSAS project management, the DCSS, and the alliance at the end of month four prior to the start of the Phase II rollout.

Cutover Month	LCSA	Region	Consortium	Approximate # of Users Converted	Cumulative # of Users Converted	Approximate % of Users Converted	Cumulative % of Users Converted
1	San Luis Obispo	5	CASES	55	55	1%	1%
2	Sonoma	2	CASES	145	245	2%	3%
	Sutter	1	CASES	45			
3			Evaluation	0	245	0	3%
4			Evaluation	0	245	0	3%

Figure 3C.13.2.4-21 Phase II Pilot Rollout Schedule - *Proposed Phase II Pilot Implementation Sequence and Schedule by LCSA.*

Phase II Non-pilot Schedule

Once the pilot implementation is concluded and the go decision is made to proceed with statewide rollout, the remaining 55 LCSAs are rolled out over a 20-month period as shown in Figure 3C.13.2.4-22, Phase II Rollout Schedule.

Cutover Month	LCSA	Region	Consortium	Approximate # of Users Converted	Cumulative # of Users Converted	Approximate % of Users Converted	Cumulative % of Users Converted
1	Tulare	4	CASES	250	665	4%	7%
	Monterey	4	CASES	130			
	San Benito	4	CASES	25			
	Calaveras	3	CASES	15			
2	Sacramento	4	CASES	275	1,085	4%	12%
	Kings	4	CASES	75			
	Santa Cruz	2	CASES	70			
3	Merced	4	CASES	125	1,425	4%	16%
	Santa	5	CASES	120			
	Barbara	3	CASES	50			
	Siskiyou	1	CASES	45			
	Madera						
4	Solano	4	CASES	200	1,695	3%	19%
	El Dorado	3	CASES	65			
	Mono	3	CASES	5			
5	Fresno	4	CASES	310	2,140	6%	24%
	Shasta	1	CASES	115			
	Alpine	3	CASES	5			
	Lassen	1	CASES	15			
6	Orange	6	ARS	715	2,970	9%	33%
	Placer	3	CASES	85			
	Mariposa	3	CASES	10			
	Tuolumne	3	CASES	20			
7	No LCSAs scheduled			0	2,970	0%	33%
8	Yolo	1	CASES	80	3,145	2%	35%
	Humboldt	1	CASES	80			
	Trinity	1	CASES	10			
	Sierra	3	CASES	5			
9	Contra Costa	2	CASES	250	3,580	5%	40%
	Butte	1	CASES	140			
	Plumas	1	CASES	15			
	Tehama	1	CASES	30			

Cutover Month	LCSA	Region	Consortium	Approximate # of Users Converted	Cumulative # of Users Converted	Approximate % of Users Converted	Cumulative % of Users Converted
10	San Bernardino	5	CASES	460	4,085	5%	45%
	Amador	3	CASES	15			
	Del Norte	1	CASES	30			
11	San Joaquin	4	CASES	200	4,505	5%	50%
	Stanislaus	2	CASES	165			
	Modoc	1	CASES	10			
	Nevada	3	CASES	45			
12	Santa Clara	2	CASES	335	4,915	5%	55%
	Napa	2	CASES	40			
	Lake	1	CASES	35			
13	Ventura	5	CASES	265	5,230	3%	58%
	Inyo	3	CASES	10			
	Mendocino	1	CASES	40			
14	No LCSAs scheduled			0	5,230	0	58%
15	San Diego	6	ARS	700	6,255	12%	70%
	Alameda	2	CASES	245			
	Yuba	1	CASES	30			
	Marin	2	CASES	50			
16	Riverside	5	CASES	475	6,755	5%	75%
	Glenn	1	CASES	15			
	Colusa	1	CASES	10			
17	Kern	5	CASES	250	7,160	5%	80%
	San Mateo	2	CASES	115			
	Imperial	5	CASES	40			
18	San Francisco	2	CASES	140	7,300	1%	81%
19	No LCSAs scheduled			0	7,300	0%	81%
20	Los Angeles	6	ARS	1,695	8,995	19%	100%

Figure 3C.13.2.4-22 Phase II Rollout Schedule - *Proposed Phase II Implementation Sequence and Schedule by LCSA.*

Again, the application of our scheduling approach may result in many possible outcomes. It is the alliance's intention to work closely with DCSS and CCSAS project management as part of the Phase II planning task to determine the actual, desired rollout sequence given the many factors, such as regionalization and minimizing county disruption, that can influence the outcome.

3C.13.2.4.3 SDU Transition Approach

This section of the alliance's approach provides additional detail regarding the transition of financial functions from the local child support agencies to the Statewide Disbursement Unit.

Phase I

The alliance proposes that the State implement the Collections functions of the SDU as part of the implementation of the Version 1 functionality that will be deployed during Phase I of the project. To support this transition the alliance has scheduled the design and development of the interface with the SDU for Phase I of the project. This interface will support the transfer of collections information from the SDU to the CCSAS CSE database and system.

The alliance proposes that the following financial functionality transitions occur during Phase I of the project.

- Redirect payments from employers, other states, individual and other payment sources to the SDU payment address.
- Transfer payment receipting responsibility to the SDU. LCSAs continue posting the occasional "walk-in" payment.
- Transfer responsibility for payment allocation to the CCSAS CSE system.

The responsibility for distribution of support payments will remain with the LCSAs during Phase I while the SDU takes responsibility for disbursement

To promote a smooth transition and avoid disruption of payments and services, the alliance recommends that the redirection of payments to the SDU also be phased. The alliance has studied a number of alternatives for phasing the redirection of payments and recommends that an employer-based approach be used. This allows payments from an employer to be redirected to the SDU at once, simplifying the redirection for the employers, the SDU and the LCSAs.

The diagram presented below depicts the flow of financial processes after the Phase I SDU capabilities have been implemented.

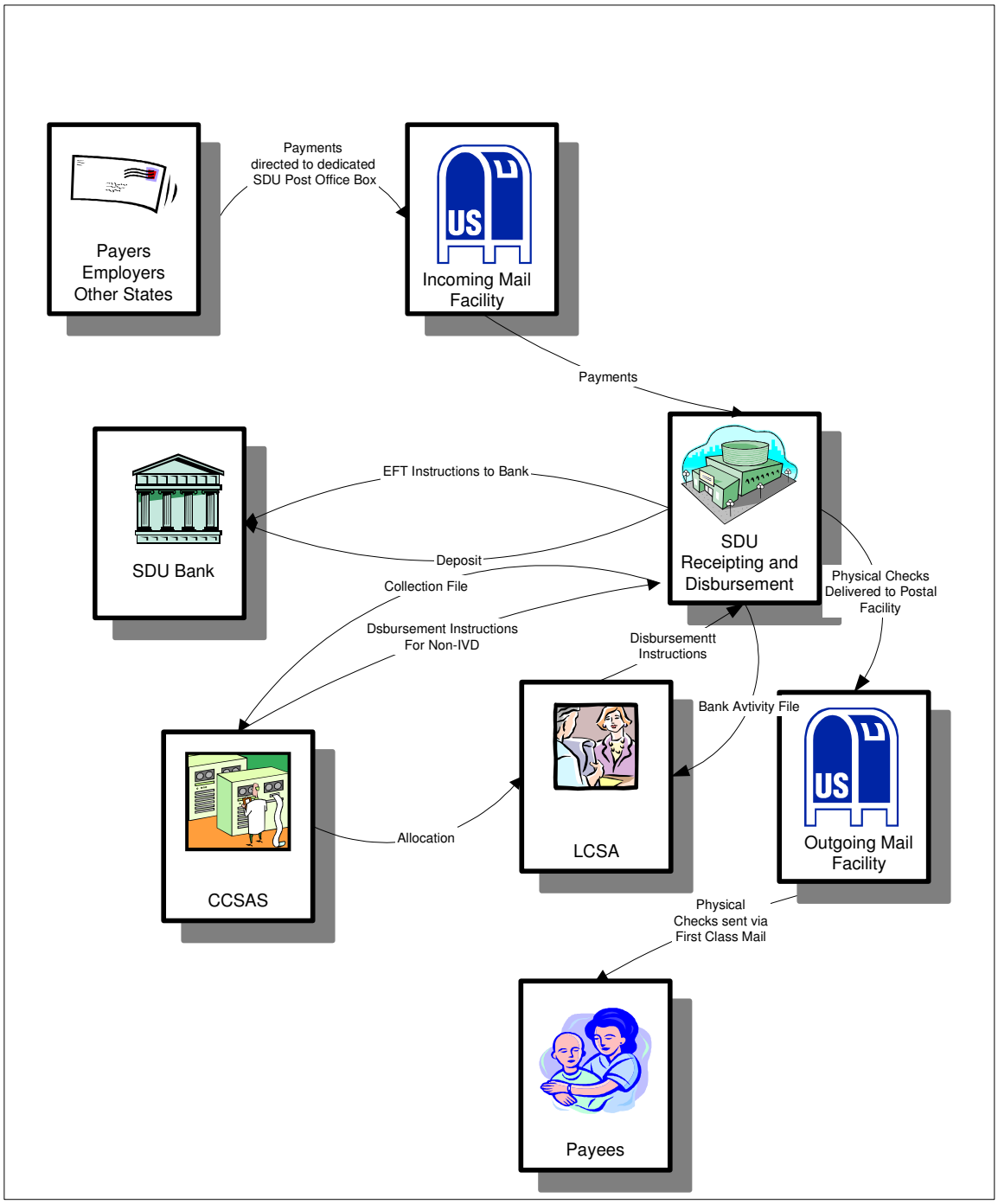


Figure 3C.13.2.4-23 Phase I SDU Workflow Transition - *This graphic depicts the flow of payment information after implementation of Phase I functionality.*

As the diagram indicates employers, other states, and individuals send payments to the new single point of collection the SDU provides. The SDU vendor receives and processes these collections.

- The SDU deposits funds into its financial institution and creates a collections file that is forwarded to the CCSAS CSE system.
- The CCSAS CSE system receives the collection file, performs payment allocation and forwards this information to the LCSAs.
- The CCSAS CSE system forwards the allocated payments to the LCSAs, which receive and process these incoming collections.
- The CCSAS CSE system also forwards disbursement instructions to the SDU for Non-IV-D orders.,
- The LCSAs, after receiving the collection information, process the support payments (distribution), and sends disbursement instructions to the SDU.

Figure 3C.13.2.4-24 [RESERVE]

Phase II

The diagram presented below depicts the flow of financial processes after the Phase II SDU capabilities have been implemented.

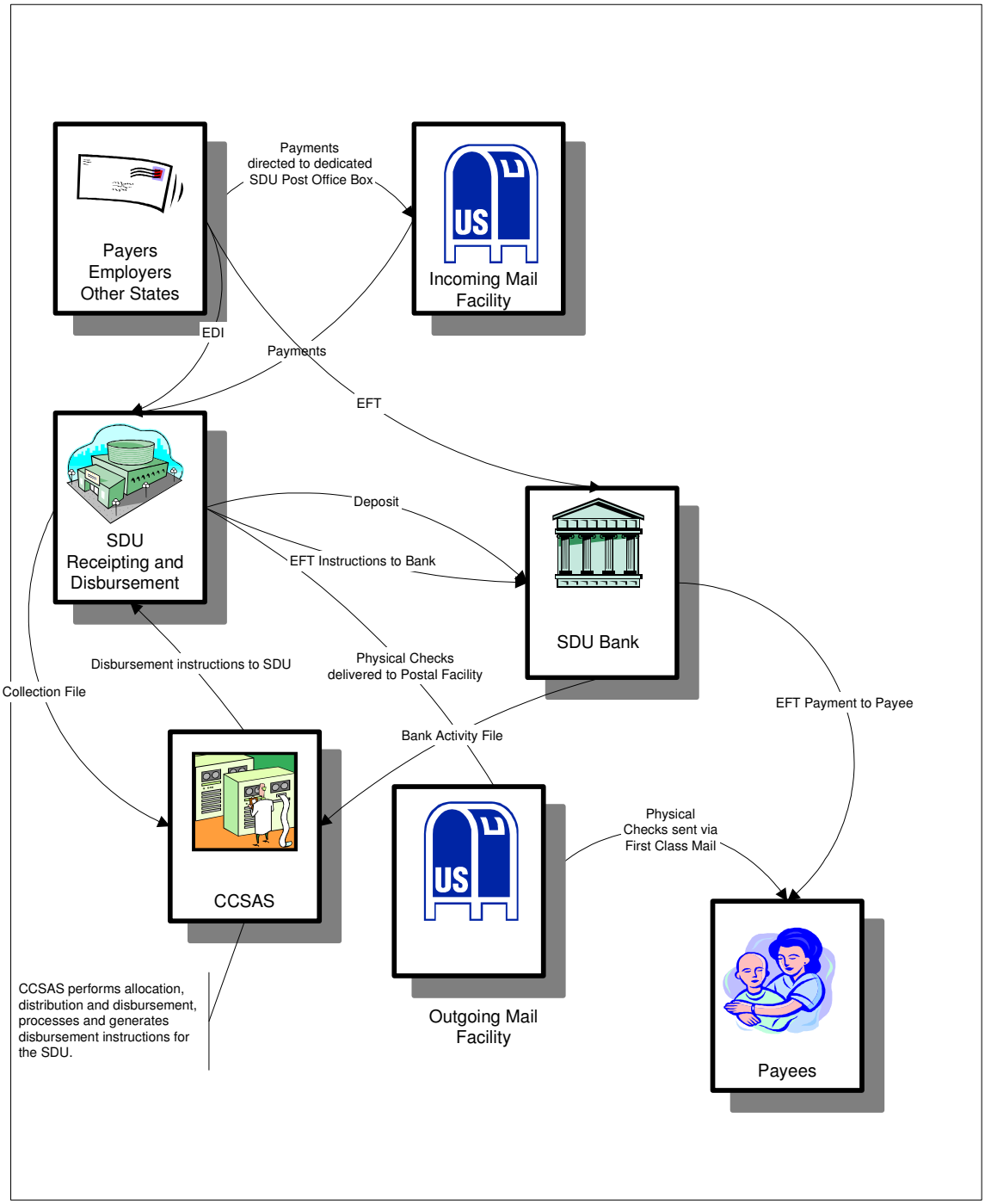


Figure 3C.13.2.4-25 SDU Workflow Transition Phase II - *This graphic depicts the flow of payment information after implementation of Phase II functionality.*

As the diagram indicates employers, other states, and individuals send payments to the single point of collection implemented during Phase I. The SDU vendor receives and processes these collections.

- The SDU deposits funds into its financial institution and creates a collections file that is forwarded to the CCSAS CSE system.
- The CCSAS CSE system receives the collection file, performs payment allocation, distribution and creates disbursement instructions for the SDU.
- The CCSAS CSE system forwards the disbursement instructions to the SDU, which receives the instructions and generates either payment checks or transfers.

3C.13.2.4.4 [RESERVE]

3C.13.2.5 On-Site Support

The goal of On-Site Support is to support child support system users through the transition to the new system and reduce the impact on county operations. On-site support matches system experts with LCSA users in the field during and after cutover to CASES and to the new CCSAS CSE best of breed system.

On-site support is one way the alliance approach readies users for the new CSE systems and supports them in system use. Other activities that prepare users for system use are orientation and user training. To support users on an ongoing basis, we provide user reference materials (e.g., On-Line Help and FAQs) and access to the Help Desk.

Our approach provides pre- and post-implementation assistance during both Phase I and Phase II of the CCSAS CSE project. Prior to cutover in each phase, on-site support staff work with State staff to prepare for system cutover and perform system set-up activities. After cutover, the focus of on-site support is to assist users through the transition to the new system. Finally, the responsibility for supporting users eventually shifts to "superusers" provided by the State in the LCSAs. In Phase I, consortium, State, and LCSA staff that have expertise in the CASES application provide the majority of support. In Phase II, alliance staff provides the bulk of the support, with increasing involvement on the part of superusers in the LCSAs.

3C.13.2.5.1 On-Site Support Approach

The alliance approach is centered on the use of superusers in the LCSAs who serve as a local point of contact for users' questions and problems. As part of the planning process in each county, the alliance works with superusers to define the county's user question and issue resolution process. Depending on the county organizational structure, superusers could be peers, supervisors, or members of a specialized team such as a county-located help desk.

The following section describes the major on-site support activities by the transition stage in which they occur:

- Pre-Cutover: Setting Up for Site Transition
- Post-Cutover: Providing On-Site Support
- Ongoing Support: Transitioning Responsibility to Site Superusers and the Help Desk

3C.13.2.5.1.1 Pre-Cutover: Setting Up for Site Transition

During the pre-cutover stage of transition, on-site support staff work with State staff to complete set-up activities. Part of these preparations involve completing prior system closeout activities and conducting a readiness test that verifies LCSA staff has access to the appropriate application functions.

Role of On-Site Support Staff During Pre-Cutover Period
On-Site Support Preparation
Identify site superusers.
Define support model (e.g., the super user contact for help during initial period of on-site support and on an ongoing basis).
Develop schedule for post-cutover on-site support.
Orient county superusers to application and support procedures.
Prepare transition approach for shifting on-site support responsibility to site superusers and the Help Desk.
Site Set-Up
Create user profiles, set up user accounts and passwords, populate data tables, and direct print requests for users.
Perform readiness test.

Figure 3C.13.2.5-1 Role of On-site Support Staff During Pre-Cutover Period - *On-site support staff work with LCSA staff to complete set-up activities.*

3C.13.2.5.1.2 Post-Cutover: Providing On-Site Support

During the initial post-cutover period, the volume and complexity of user issues tend to be highest. To address this level of demand, alliance staff and superusers in the LCSAs work together to:

- Answer user questions related to the use of the system in day-to-day operations
- Aid in the resolution of system problems
- Track questions and issues

Issues and application problems are escalated to the appropriate Help Desk (either CASES in Phase I or CCSAS CSE in Phase II), as necessary. Additionally, the on-site support team will notify the Customer Service Support Center of conversion problems that may impact customer service, so that they can be prepared to handle customer questions appropriately. Questions related to policies or county procedures continue to be handled in the same manner as before by DCSS and/or the appropriate LCSA staff.

Role of On-Site Support Staff During Post-Cutover Period
Provide guidance related to use of the system for daily job functions (“How Do I”?).
Identify and report application problems related to the new CSE system.
Identify and monitor implementation issues arising in county and assist in their resolution.
Act as liaison between the project and county users to communicate important information on workarounds and system upgrades.

Figure 3C.13.2.5-2 Role of On-site Support Staff During Post-Cutover Period – *LCSA superusers and alliance staff work together to meet the volume and complexity of user issues during the initial post-cutover period.*

3C.13.2.5.1.3 Ongoing Support: Transitioning Responsibility to Site Superusers and the Help Desk

As the period of alliance-provided on-site support draws to an end, superusers in the LCSAs take on more of the responsibility as the first point of contact for questions and issue resolution. The objective of this slow shift of responsibilities is to effectively transfer knowledge to superusers so they become proficient in the new system's business workflow and system functionality prior to taking on responsibility for all on-site support.

Role of On-Site Support Staff During Transition to Ongoing Support
Support the handoff of responsibility for support to site superusers and Help Desk.
Publicize methods for contacting the Help Desk.

Figure 3C.13.2.5-3 Role of On-site Support Staff During Transition to Ongoing Support – *On-site support responsibilities are transitioned to superusers in the LCSAs.*

3C.13.2.5.2 Duration and Level of Post-Cutover On-Site Support

The alliance drew upon its experience with CSE implementations in California counties, New Mexico, and Wyoming to create a baseline staffing model for on-site support that can be applied to all site implementations. This model provides for up to two months of support after cutover in each implementation site, with the number of staff resources determined by the number of users and the timing of the implementation. The model also adjusts the source of on-site support staff (either alliance staff or State staff) to provide users with support from staff that has the greatest expertise on the particular system being implemented. The baseline staffing model is based upon the following considerations:

- ***Duration of on-site support:*** On-Site Support will be provided for one month for the CASES Conversions (Phase 1). The Phase II will remain two months. This duration allows for the support of month-end financial processing and certain workflow triggers that may be invoked at month end. On-site support staff begins rolling off the site following the first month after cutover (updated per CR-00069).
- ***Level of on-site support:*** The level of support is based on the number of users at the site, and is based on providing support for up to 10,000 users statewide. The ratio applied to the number of users varies according to the implementation Phase (I or II) and the timing of the implementation. For Phase I, the level of support remains constant throughout the implementation phase as the system being implemented, CASES, has already been piloted and is in operation in many counties throughout the state. For Phase II, a higher level of support is provided during pilot and early implementations to account for the increased effort associated with implementing a new system.
- ***Division of responsibility:*** alliance or State staff that have the greatest expertise with the system being implemented provides on-site support. The on-site support model also incorporates the use of site superusers who serve as the first point of contact for questions and issues.

3C.13.2.5.2.1 On-Site Implementation Support Plan (CDL TM 061)

The alliance prepares a On-Site Implementation Support Plan with site-specific information prior to beginning implementation activities in each phase. This plan covers:

- The classification of staff required
- The number by classification
- The number of State, LCSA, and alliance staff
- The length of support by classification
- The differentiation between on-site support and Help Desk support (which will be further defined by each LCSA)
- Roles and responsibilities

3C.13.2.5.2.2 Phase I On-Site Support Model for Implementation of CASES

On-site support for the 14-county rollout of CASES in Phase I is provided primarily by Consortium or State staff, with assistance from the alliance. This approach leverages the CASES system knowledge already possessed by State staff, utilizing alliance staff that have experience providing on-site support. Figure 3C.13.2.5-4 outlines the on-site support staff ratios and division of responsibility between the alliance and State staff for Phase I.

Phase I On-Site Support Model		
Support level ratio ***(number of users: number of on-site support staff)***		
	Month following cutover	Second month after cutover
Phase I counties	25:1	0 (users updated per CR-00069)
Division of responsibility		
alliance	25%	
State	75%	

Figure 3C.13.2.5-4 Phase I On-Site Support Model - *On-site staffing ratios and staffing responsibilities for Phase I.*

Because CASES is already in production in several counties, it is assumed that the support level ratio does not need to be higher for the pilot counties, or for the initial counties that will go live just after the pilot. For those counties that have fewer users than indicated in the staffing ratio, a minimum of one alliance person is available to provide on-site support.

3C.13.2.5.2.3 Phase II On-Site Support Model for Implementation of Version 2 CSE System

For Phase II, the on-site support model takes into account that we will be introducing a new system into production. Accordingly, the on-site support level ratios are higher for the pilot and the initial set of counties after pilot, in order to account for the increased level of effort needed to support a transition to a new system. Additionally, the division of responsibility moves to the alliance providing the majority of on-site support. The alliance staff will have the greatest knowledge of the new Version 2 CSE system based on our experience with building the system. Figure 3C.13.2.5-2 outlines the on-site support staff ratios and division of responsibility between the alliance and State staff for Phase II.

Phase II On-Site Support Model		
Support level ratio ***(number of users: number of on-site support staff)***		
	Month following cutover	Second month after cutover
Pilot	10:1	20:1
Initial group of counties (those within 6 months after pilot)	15:1	30:1
Final group of counties	30:1	60:1
Division of responsibility		
alliance	75%	
State	25%	

Figure 3C.13.2.5-5 Phase II On-Site Support Model - *On-site staffing ratios and staffing responsibilities for Phase II.*

3C.13.2.5.3 Staffing for On-Site Support

To effectively support users during the period of on-site support, alliance staff will have the specific skills required to perform the duties of the position:

- Knowledge of the prior system and new application functionality
- Customer service skills
- Strong understanding of the child support program
- Familiarity with the county organization and general county business processes

Further, as the rollout continues, information and lessons learned from each LCSA implementation are captured in the Post-Implementation Review Report (CDL TM 039) and transferred to other implementation staff. This allows us to learn from the on-site support experiences of each implementation.

During peak periods, providing a sufficient number of qualified staff to fill on-site support positions is critical to the success of the implementation effort. The alliance on-site support approach draws from the expertise of other project teams as the work of these teams begins to ramp down and staff become available. This approach lends expertise to those areas where it is needed and levels overall project staff resource requirements. Staff that may be utilized include staff from application development, help desk, training, and other implementation staff.

3C.13.2.5.4 [RESERVE]

3C.13.2.6 Local Interfaces

This section describes the process of identifying and establishing connections where standard interfaces have been developed for the purpose of exchanging information with a local county agency. The local interfaces for the CCSAS CSE system will be implemented locally, but will be built on a standard interface structure across all counties. This section does not describe the process of designing and developing the interfaces, as these are functions of the application development team.

For Phase I, the alliance utilizes the existing CASES welfare and auditor/controller interfaces. The Business Partner performs the activities necessary to verify local interface connections.

For Phase II, the CCSAS CSE system will provide an automated interface to three local entities: the County Court, the County Welfare Department (Temporary Aid for Needy Families or TANF), and the County Recorder's Office. The alliance proposes to develop a single common interface standard for each of these automated interfaces. The standard welfare interface will be developed for use with the four welfare consortia systems.

Figure 3C.13.2.6-1 summarizes the electronic local interfaces to be addressed in CCSAS CSE.

CSCAS CSE Version 2 Electronic Local Interfaces	
Local County Agency	Interface Description
County Courts	California has 58 superior courts that hear and adjudicate child support cases. Those cases involve the establishment of paternity (with the exception of administrative or stipulated paternities) and child support orders, medical support, and modifications to existing orders and enforcement actions including liens and civil and criminal complaints. The CCSAS CSE system will interface with the courts to obtain the court calendar and available times to schedule child support cases.
DSS Welfare Department (TANF)	The California State Department of Social Services' welfare program provides services that include issuance of food stamps, cash aid, and medical and foster care to needy children and families in the State of California. The CCSAS CSE system will automatically receive and process referrals from the State's TANF, IV-E, and Title XIX agencies. CCSAS CSE will receive financial grant information (amount of monthly grant), direct support payment, good cause, recipient and absent parent information, cooperation status, case updates, case discontinuance, and general case information. Also, CCSAS CSE will provide the TANF agency with information to assist in eligibility determination, including the amount of current support, collections, distribution, and direct payment information for public assistance cases.

CSCAS CSE Version 2 Electronic Local Interfaces	
County Recorder's Office	The Recorder's Office is responsible for the recording of deeds, mortgages, decrees of court, and leases affecting title to real property in the county. The office also maintains vital statistics, including births, deaths and marriages within the county. The CCSAS CSE application will interface with the County Recorder's Office to exchange information on locate and to register liens and levies.

Figure 3C.13.2.6-1 Summary of Local County Interfaces - *This table provides a list of the CCSAS CSE automated local interfaces for Version 1. The Local Interface Team is responsible for the coordination of the activities necessary to establish electronic connections.*

3C.13.2.6.1 Local Interface Approach

As part of its implementation effort, the alliance will work with the State to identify and establish interface connections between the new statewide child support system and the LCSA. We work with the LCSAs to help them establish local interface connections given the standard interface provided.

The four welfare systems shown in Figure 3C.13.2.6-2 are assumed to be the only four systems in production at the time of implementation. They represent the four welfare systems that are supported by the new statewide child support system and for which electronic interface connections may be established through the common, standard interface.

Welfare System	County Systems
1. C-IV	Merced, Riverside, San Bernardino, and Stanislaus
2. CalWIN	Alameda, Contra Costa, Fresno, Orange, Placer, Sacramento, San Diego, San Francisco, San Luis Obispo, San Mateo, Santa Barbara, Santa Cruz, Santa Clara, Solano, Sonoma, Tulare, Ventura, and Yolo
3. ISAWS	Alpine, Amador, Butte, Calaveras, Colusa, Del Norte, El Dorado, Glenn, Humboldt, Imperial, Inyo, Kern, Kings, Lake, Lassen, Madera, Marin, Mariposa, Mendocino, Modoc, Mono, Monterey, Napa, Nevada, Plumas, San Benito, San Joaquin, Shasta, Sierra, Siskiyou, Sutter, Tehama, Trinity, Tuolumne, and Yuba
4. Leader	Los Angeles

Figure 3C.13.2.6-2 The Four California Welfare Systems - *This depicts the four welfare systems that are or will be in use in the State of California at the time of the CCSAS CSE implementation.*

While the Local Interface Team works with the LCSAs and local county agencies to implement these interfaces, the scope of their work is limited to the verification of interface connections. Any modification that might be necessary for the local county agency's system to establish a connection with the statewide child support enforcement

system becomes the responsibility of the State. Similarly, during Phase I, any modifications needed to make CASES properly interface with local IV-A and auditor/controller systems will be the responsibility of the State.

Staff assigned to the Local Interface Team perform the following tasks as part of the LCSA's transition to the new child support enforcement system:

- Identify and define local interface needs
- Monitor local interface work
- Verify connections

The Local Interface Team staff is organized and assigned to the LCSAs based on the welfare interface used by the county, as it will be the more complex of the three electronic interfaces to be established in this project. We believe the more expertise held by the Local Interface personnel in the welfare systems and interface requirements, the better they are equipped to work with welfare consortia and county staff to establish interface connections and assist with readiness testing. The efficiencies gained through this staffing approach reduce the time necessary to establish and test the local interfaces. The incorporation of lessons learned at this level increases the quality of the work and limits the number of issues and amount of rework.

Identify and Define Local Interface Needs

Members of the Local Interface Team will work with the LCSA and local county agencies to identify and document interfaces necessary to perform case and financial management functions. Where electronic interfaces are available, the interface team will work with the corresponding county agency to provide interface specifications and to define the timing, frequency, transport mechanism, media, and other considerations that might be necessary to establish the connection.

Where no electronic interfaces are available, the Local Interface Team will work closely with the LCSA and alliance change management staff assigned to the implementation team to document the interface as part of the "To-Be" model. This supports the definition of the necessary letters, forms, terminals, CDs, or other media available for direct access.

Monitor Local Interface Work

As part of the LCSA's Pre-Implementation Readiness Assessment (TM 038), the State documents each of the local interfaces slated for implementation. The assessment lists the steps necessary to implement each interface, validate connectivity, and perform an interface readiness assessment. The alliance monitors these steps as part of the regular pre-implementation readiness assessment process.

Verify Connections

Prior to cutover, the Local Interface Team works closely with local county agencies and LCSAs, as appropriate, to verify interface connections and performance. During this period the local interface team monitors progress and works to resolve any issues that

might exist. Coding and unit testing are conducted in the test environment, which is isolated from the production and development environments. This is necessary so that test activities do not interfere with development or production. Once interface connections are established, the local interface team assists the local county agency and LCSA to verify the connections and the exchange of information. Interface verification is intended to determine if the interface connections, as established, support the requirements and specifications of the interfaces as defined for a particular LCSA and/or county.

3C.13.2.6.2 [RESERVE]

3C.13.2.7 Conclusion

The alliance approach to Transition Management demonstrates how our carefully planned transition approach, proven implementation methodology, and management approach can be used to successfully implement the next generation child support enforcement system in an environment that is constrained by disparate systems, complex business rules, number of locations, and changes in the way of doing business.

Implementation of the new system is constrained by disparate systems, complex business rules, number of locations, and changes in the way of doing business.

To accomplish this, the alliance uses an approach to transition management that supports a proven implementation approach. Our implementation approach employs seven disciplines: Implementation Management, Change Management, Conversion, Hardware/Software Installation, User Training, Local Interfaces, and On-Site Support, and specialized skills to address aspects of system implementation to provide successful delivery and mitigate and manage risk. Of these seven disciplines, four are covered under this transition management section.

As discussed in this section, our Change Management approach strives to achieve uniformity while lessoning the impact of change on the programs users and customers. It was derived through lessons learned from CWS/CMS and is founded upon committed sponsorship; thorough understanding of the change; and strong communication and involvement principals. It includes a proven method for performing business process analysis that fully supports the State's goal to achieve uniformity and lessen the impact of change.

Our Implementation Management approach provides a framework for managing the entire implementation organization and for managing the implementation efforts at each site.

Our proven implementation methodology and extensive consortia system experience also mitigates and manages risk through the use of experienced staff dedicated to providing On-Site Support at each LCSA during the initial period after cutover.

Our approach to Local Interfaces applies our specialized knowledge of child support enforcement systems and implementation to a high-risk area that is often neglected. The alliance, through lessons learned from CWS/CMS, recognizes the need for dedicated staff in this area to help identify and establish local interfaces prior to LCSA cutover. These include, where applicable, the welfare interface and court interface. Our approach in this area includes the processes necessary to identify, configure, and establish interface connections, along with the methods needed to plan and execute readiness checks for each automated interface prior to system cutover.

In conclusion, we believe our carefully planned transition approach, proven implementation methodology, and management approach is the choice for the CCSAS CSE project to use in order to successfully implement the next generation child support

enforcement system in an environment composed of disparate systems, complex business rules, numerous locations, and changes in the way of doing business.

3C.13.3 User Training

3C.13.3.1 Introduction

The alliance child support training solution reuses existing CASES training components in delivering CASES training, and leverages the "anytime, everywhere" capability of web-based e-learning to deliver CCSAS CSE Version 2 application training during Phase II. CCSAS CSE Version 2 training is delivered in both instructor-led and coach-assisted formats based on the complexity of the topic. The alliance training approach prepares users for cutover to CSE and provides materials for ongoing user reference.

Training users, including LCSA and DCSS staff, on how to use the new CSE system is an integral component of the overall alliance implementation initiative. Our proposed training solution is delivered in two phases, in line with the two-phase structure of the overall project. In addition, the alliance will provide training program support for a period of two years after implementation. At the end of the two-year post implementation period, training program management will be transitioned to DCSS control.

The alliance training team is accomplished in both traditional and non-traditional methods of instruction. Members of the team have served on many successful projects where classroom and e-learning delivery methods were used to achieve the learning objectives of the user community.

The alliance has a strong reputation for providing quality training in the child support arena. Our team has provided technical assistance and training to child support professionals of various levels, including caseworkers, managers, attorneys, and judges. The alliance has delivered training on virtually every aspect of child support, including paternity establishment, child support guidelines, enforcement, PRWORA policy changes, and system requirements. The alliance brings experience with delivering child support training in several other states.

3C.13.3.2 Approach

3C.13.3.2.1 Overview of Training Strategy

During Phase I, our training strategy addresses the training needs of the 21 counties converting to the CASES system. During Phase II, our training strategy addresses the training needs of the 58 counties before cutover to the new CCSAS CSE Version 2 system. Figure 3C.13.3-1 presents an overview of our training strategy, highlighting and comparing key aspects of the approach for each phase.

Phase I - CASES Implementation		Phase II - Best of Breed CSE Application Implementation
CASES Statewide Services	***Child Support Application***	**New Best of Breed CSE Application**
14	***Number of Counties to be Trained***	**58**
LCSA Staff State Staff	***Trainee Groups***	**LCSA Staff State Staff**
Reuse of existing CASES Training Program materials, methods, and delivery approach	***Training Strategy***	**Custom development of all components required to support rollout of the new statewide CCCSAS system**
Delivery by LCSA Trainers	***Strategy Characteristics***	**CCSAS application used with e-learning course modules to deliver scenario based training Instructor-Led Classroom Training Coach-Assisted E-Learning at the Desktop Learning Management System Training Database**
Sunset CASES Training Program with rollout of Best of Breed CCSAS Application	***Post-Implementation***	**Maintain and Operate CCSAS Training Program Transition to State Control at End of Two Years**

Figure 3C.13.3-1 Overview of Alliance Training Strategy - *The alliance approach requires two distinct solutions for user training.*

Overview of Training During Phase I

During Phase I, the strategy for training leverages the current methods and delivery approach that have been utilized on the CASES transitions to date.

In order to divide the responsibilities of the training activities, we explored the current training approach and placed responsibility for delivery and infrastructure with the organizations that had the most experience conducting those activities. To provide a coordinated approach, particularly as the CASES implementation effort entails a more compressed timeline than has been underway to date, the alliance will perform management and support activities. Under this approach, the alliance will produce an overall plan for CASES training rollout and scheduling.

CASES training leverages the proven approach already used in 34 CASES counties by dividing the responsibilities for management, delivery, and infrastructure between the alliance, the CASES Consortium, and the LCSAs.

For delivery of CASES user training in Phase I, our solution specifies that it continue to be performed as it is now, through a combination of the CASES consortium staff and LCSA staff. We propose to continue with the approach that San Francisco, as the lead CASES consortia county utilizes, by providing a Train the Trainers session to key LCSA staff, who then return to their home county to train the remaining LCSA users. In Phase I, infrastructure activities to support training such as training facilities and the training environment will be handled by the Consortium and LCSAs, as they are now.

In addition to converting 14 counties to CASES, our training strategy during Phase I involves the development and implementation of the Statewide Services component. For CSE Version I, the BP will have primary responsibility for the development and delivery of the training curriculum and materials. The State will support the BP in this effort. In order to capture all Version I user impacts the BP has agreed to deliver an addendum to TM064-1. The addendum will provide the same detail for the additional user impacts, as the initial delivery will have for Federal Reporting and Non IV-D Order Entry. The addendum will also provide an update to any information found in the plan as necessary. Although not finalized, the current list of high-level functionality where users will have on-line interaction with SWS is: Non IV-D Order Entry, Generate and View Reports (including Federal and On-Demand Reports), Payment Inquiry, Working Suspended Payments, Disbursement Processing, Payment Allocation Holds/Release, Payment Adjustments, IV-D Fund Reconciliation Reports, Reference Table and User Profile Maintenance, Activity Log Maintenance, CSENet-CCR functions, Locate Activity Inquiry, Case/Participant Inquiry and Payee Direct Deposit. The curriculum and associated training materials will be consistent with the agreed upon functionality identified in the SRS-1, Software Requirements. Figure 3.c.13.3-2 summarizes the proposed division of responsibility of training tasks for Phase I based on the recognition of the BP's primary author role and the State's agreement to support the BP's training curriculum and material development activities (paragraph replace per CR-00087).

Type of Activity	Activity	Responsibility
	CASES Training for 14 Counties	
Management and Support	Readiness Assessments	Alliance
	Scheduling	Alliance
	Statewide Plan	Alliance
Delivery	Selecting trainers	State
	Modifying training materials	State
	Conducting Train the Trainer sessions	State
	Conducting user training sessions	State
	Producing attendance reports (optional)	State

	Evaluation of trainees (optional)	State
Infrastructure	Determination and setup of training site (facilities and PCs)	State
	Training Database/Environment setup	Alliance
CSE Version 1 Statewide Services Training		
	Creation of training materials	Alliance with support from the State (updated per CR-00124a)
	Determining which users need materials	State
	Posting material on web site and notifying users of its availability	Alliance

Figure 3C.13.3-2 Summary of Phase I Activities and Responsibility – *Our approach divides the responsibilities for training in Phase I between the alliance and the State.*

Our CSE Version 2 training solution is custom developed using a tailored methodology for curriculum design and development. Training modules are delivered via e-learning, which is well suited to the implementation of a statewide system.

Overview of Training during Phase II

Our CSE Version 2 training solution involves the custom design and development of CCSAS CSE application training curriculum, and the use of technology-enhanced delivery methods suitable for training geographically dispersed trainees. The foundation of our solution centers on an e-learning course-module design approach. Using the CCSAS CSE application to create the training environment, the trainee is guided through a lesson by a scenario-based e-learning module defined within the context of a business process. The business processes will be based upon the new CCSAS Business Model developed by DCSS in association with the alliance Change Management team.

E-learning module delivery methods include both instructor-led classroom training and coach-assisted workplace training, depending upon the complexity of the training topic. Classroom-based training will be delivered at four different regional training centers.

During CSE Version 2 training in Phase II, up to 10,000 LCSA staff in California's 58 counties and staff at State-level sites will be trained to use the new best of breed CCSAS CSE application. Figure 3C.13.3-3 presents the features and benefits of our CSE Version 2 training strategy.

Features		**Benefits**
Methodology	Instructional Systems Design Methodology	Provides a structured framework for training needs assessment, design, development and delivery of training.
Design Techniques	Adult Learning Principles	Recognizes that training for adult learners needs to focus on effective and efficient modes of delivery, particularly for busy child support trainees.

Features		Benefits
	Scenario-Based Learning	Works in conjunction with scenario-based learning. Focuses on trainee retention by utilizing scenario-based learning techniques. Scenario-based learning allows trainees to work on real-life tasks and receive immediate feedback upon completion of those tasks.
Design Artifacts	CCSAS Business Model	Serves as the training context within which scenario exercises are developed. By adopting business processes from the CCSAS Business Model, the trainees conduct exercises using business processes that reflect operational flows after cutover to the new system.
Delivery Tools	E-Learning	Utilizes web-based training, delivering content over the secured CCSAS CSE project web site. Our e-learning approach is particularly suited to delivering training to large, geographically dispersed audiences. The secure intranet allows easy distribution of updated e-learning modules, so that trainees have access to the most recent versions available. With e-learning, trainees receive consistent training regardless of delivery method. Courseware is unified, distributed from a single source, and used in both classroom and coach-assisted environments. The e-learning courses are also available continuously, so staff is not constrained by only having one opportunity to take the learning module.
	CCSAS CSE Application	Utilizes the CCSAS CSE application as a training tool. Trainees work through scenarios within each course module using the CSE training application. This promotes an accelerated achievement of proficiency using the new system on the job.
	Learning Management System	Manages the e-learning course modules in terms of version control, distribution over the secure intranet, and maintenance. The LMS also tracks attendance and trainee performance.
Delivery Approach	Just in Time Training	Delivered as close as possible to cutover, providing for short-term reinforcement of skills learned during training. For most counties, training occurs in the five weeks prior to cutover. Larger counties' training occurs in the eight weeks prior to training.
Delivery Methods	Instructor-Led Classroom Training Coach-Assisted On-Site Training	Varies depending upon what best fits the training content. The alliance's e-learning application provides the foundation for both instructor-led classroom training at the regional training centers and coach-assisted training at the implementation sites. Instructor-led training covers financial topics, providing trainees with an opportunity to clarify their understanding of complex functions and transactions through discussions with peers and alliance staff. The coach-assisted format provides training at the workplace using the e-learning courses for case management topics. Coach-assisted training supports learners who learn at varying paces. Implementing a strategy where the majority of training is delivered in the coach-assisted format greatly reduces the travel time and costs for LCSA trainees since they will not be traveling to training centers for their entire training.

Figure 3C.13.3-3 Features and Benefits of CSE Version 2 Training Strategy – *The alliance approach to CSE Version 2 training is innovative and enables the leveraging of the solution into the future.*

Overview of Post-Implementation Training

During the two-year post-implementation period of the project, the alliance will maintain and operate the CCSAS CSE training program, including e-learning application maintenance, course content upkeep and release management, and administration of one regional training center. To accomplish the smooth transition of the program to DCSS, the alliance proposes that DCSS work with CCSAS project and alliance staff from the start of the two-year period. This enables DCSS to become proficient in the maintenance and upkeep of course content and e-learning modules prior to taking full ownership of the program.

The details of our approach are organized into the following sections:

- Phase I training
- Phase II training
- Post-implementation training

3C.13.3.2.2 Phase I Training

During Phase 1, two classes of users require training. First, staff in the 14 LCSAs converting to CASES receive training. Secondly, staff affected by changes for the CSE Version 1 Statewide Services component receive training materials.

CASES Training

To leverage existing expertise on the CASES system, CASES training will be conducted as a collaborative effort between the alliance and the State. We propose building on the successful CASES training rollout by utilizing a proven formula that lessens the cost to the State. The majority of the activity is performed by the State, with the alliance training staff playing a supporting role. Alliance staff will be responsible for the following management and support activities:

- Readiness assessments
- One overall rollout plan
- Scheduling

The State will be responsible for the delivery and infrastructure activities: training of trainers, training materials, user training delivery, training database setup, and training sites.

Readiness Assessments

To adequately plan for the delivery of CASES training, we will conduct a series of pre-assessment and readiness assessments surveys as part of the implementation assessment that will provide critical information to the planning process. The alliance training team coordinates its preparation activities through the key events that take place for each implementation site:

- ***Pre-Assessment Survey:*** A section of the Phase I Pre-Assessment Survey is designed to gain a better understanding of the training needs of the LCSAs. Questions relate to the number and classifications of staff to be trained; whether the county has existing training facilities, and whether the county has existing trainers or other highly skilled users who will participate in the Train the Trainer program. We will pass the results of these surveys to the CCSAS project and LCSAs.
- ***Pre-Implementation Readiness Assessment (CDL TM 038):*** The training section of the Phase I Pre-Implementation Readiness Assessment, which is conducted at specific time intervals over the implementation period for each LCSA, determines whether the LCSA is equipped for training and whether the users are trained so that cutover may occur.

Plan

The alliance will develop a training rollout plan for the counties converting to CASES. The plan will be at a higher level than the User Training and Train the Trainers Plans developed for CSE Version 2 training. Since the alliance is only taking responsibility for management-type activities for CASES training and because the basic approach and activities are already in use by the Consortium, we do not plan to address all training related CDLs.

CASES Training Delivery

Train the Trainers

The State will deliver Train the Trainers courses to LCSA staff instructors at appropriate intervals prior to cutover. The alliance will provide scheduling support for Train the Trainers sessions for the CASES trainers in accordance with the level of effort outlined in the detailed task plan. The State will identify the appropriate personnel to act as trainers and to attend the CASES Train the Trainers sessions.

CASES Application User Training

We assume that CASES Application User Training will be delivered to LCSA staff in the 14 counties converting to CASES by LCSA trainers. The alliance will provide scheduling support for the training sessions associated with the 14 county rollout. We will work with the LCSA training liaisons to determine which staff attend which sessions, and notify staff of their training schedule via email or a letter. The schedule will include the location of the training, the date and time of the training, the name of LCSA trainer(s), and which county staff are scheduled to participate in the training.

Training Materials

We assume that the existing training materials currently utilized by the CASES Consortium can continue to be used for CASES training with little to no modification.
Training Attendance Reports (Optional)

Each LCSA may develop a course attendance tracking method to produce attendance reports for Phase I training. The LCSA trainers then record trainee attendance at each session.

CASES Training Facilities and Database

We assume that the Consortium continues to provide classrooms for the Train the Trainers sessions for the CASES trainers. We also assume that LCSAs will identify facilities for the training of the their own staff. The database setup for training continues to be performed by the Consortium lead county.

CSE Version 1 Statewide Services Training Materials

Of the several items being implemented during the CSE Version 1 Statewide Services portion of Phase I, only two require user instruction. As both of these components are used by a targeted group of users, we will provide self paced training instructional material for these users. The alliance will create these training materials and make them available to the relevant staff via the CCSAS CSE project web site. Select LCSA staff will receive materials on how to enter data into the web-enabled screens for federal reporting. Select DCSS staff will receive materials on how to produce and read the federal reports. Select DCSS staff also will receive materials on entering non IV-D orders into the State Case Registry. The Figure 3C.13.3.4 summarizes the training material to be developed for the CSE Version 1 Statewide Services in Phase I.

CSE Version 1 Statewide Services Component	Instructional Topics	Audience
Federal Reporting	How to utilize web enabled data screens to enter each county's data totals.	LCSA staff
Federal Reporting	How to produce reports and an introduction to summary views.	DCSS staff
Non IV-D Orders	How to enter non IV-D orders into the State Case Registry.	DCSS staff

Figure 3C.13.3.4 CSE Version 1 Statewide Services Training Materials – *Minimal new training material will be required for CSE Version 1 training.*

The materials will be posted to the CCSAS CSE project web site so that the chosen trainees may access them once the alliance notifies them that they are available.

3C.13.3.2.3 Phase II Training

Phase II is centered on the design, development and delivery of training related to implementation of the new statewide CCSAS CSE Version 2 application. Three levels of user classes are described in SCP Exhibit 3H: Local, State, and Public. The training addressed in this section impacts the Local (LCSA) and State (DCSS) user classes. This section describes the details of our CSE Version 2 approach to training, and is organized by the two course types of Application User Training and Technical Support Training.

Application User Training

Scenario based e-learning and the use of technology enhanced delivery methods anchors our approach for Application User Training. Instructional Systems Design provides the foundation for the development of the training approach.

Instructional Systems Design

Instruction for the CCSAS CSE Version 2 and post-implementation user training will be designed, developed, and delivered using the Instructional Systems Design (ISD) methodology. In general, the ISD methodology consists of the following phases:

- Analyze. Audience, job performance, and task analysis of the learning domain, resulting in the definition of a clearly stated goal and the measurable performance objectives needed to reach this goal. Curricula are tied to this goal and objectives.
- Design. This stage involves using the assessment and training requirements data from the Analyze stage to execute a strategy for developing the instruction.
- Develop. Modular curricula are built based on the analysis goal and objectives. Content is derived through subject matter experts (SMEs).
- Implement. The implementation phase includes delivery of the instruction to audiences via the selected system.
- Evaluate and Improve. This stage measures the effectiveness and efficiency of the instruction. It includes evaluation of the instruction and a feedback mechanism for continuous and iterative improvement of the instruction.

Within this document, the ISD phases describe all aspects of our CCSAS CSE Version 2 User Training approach.

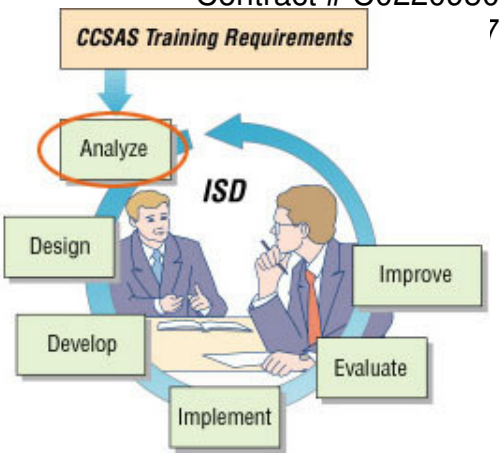
Analyze

The **analyze** stage is the foundation for other phases of instructional design. The purpose of this stage is to identify the user groups and determine what these user groups must know or do on the job. For the CCSAS CSE project, three areas fit into this stage:

- Readiness assessment
- Training plans
- Understanding the functional architecture of the system

Readiness Assessment

To provide comprehensive training that readies the CCSAS CSE project for use of the CCSAS CSE application, the alliance will conduct a series of readiness assessments using input from LCSA staff. The readiness assessment tools described in Phase I – the Pre-Assessment Survey and the Pre-Implementation Readiness Assessments -- are also used in Phase II.



To tailor user training, we need to understand the training needs, current skill set, and job responsibilities of the user classes. We must understand the users' level of comfort with navigation of an automated system, knowledge of computers in general, and level of child support expertise. We assume that the trainees have a basic working knowledge of a Windows desktop environment, Internet browsers, and basic MS Office functionality. The assessment tools help us achieve this understanding and validate our assumptions. Further insight into the new, standardized business practices of the various counties, especially case processing, is also needed to craft beneficial training for the user classes. We will coordinate our training efforts with the CCSAS Business Model in order to achieve this insight.

For Secondary System Users such as the courts and public assistance offices, and for Third Party System Interface Users such as employers and financial institutions, we will work with the CCSAS Project Transition Management Section (TMS) to determine the best training mechanism. At this point, we have not addressed the specifics of the effort in our scope, and therefore it will be handled through the Change Request process as necessary.

Plans

Phase II Train the Trainers Plan (CDL TM 062)

The alliance will develop a *Train the Trainers* Plan for CSE Version 2 training that contains a strategy for rolling out Train the Trainer courses to staff selected by each of the 58 counties. The LCSA trainers can then supplement alliance trainers and coaches by lending assistance in the classroom and coach-assisted phases of training, providing additional training attention to users in that county. (However, in our staffing assumptions, we have not assumed their participation.) The LCSA staff that participates in Train the Trainer classes could also serve as "superusers" of the best of breed CCSAS CSE application within their home LCSA. Superusers provide support to other users in the county during the post-cutover period, along with the alliance on-site support staff. The LCSA trainers and superusers are of great benefit to the counties that choose to provide them because they will enhance the overall resources available to assist their coworkers. The *Train the Trainers* Plan also will contain a section addressing the formal training preparation session for alliance training staff.

Specifically, the CSE Version 2 *Train the Trainers* Plan will address:

- Activities
- Milestones
- Roles and responsibilities of alliance staff, LCSA staff and CCSAS project staff
- Planning for training materials and applications
- General timelines

CSE Version 2 User Training Plan (CDL TM 064)

The User Training Plan is a 'living document' that is periodically updated and produced at 6-month intervals. The User Training Plan for Phase II describes the following;

- Overall purpose and goal of the CCSAS CSE best of breed application training
- Roles and Responsibilities
- Readiness Assessments
- Approach for curriculum preparation
- Planned delivery method
- Trainee audience groups
- Schedule of training to be delivered during Phase II implementation
-

The plan will provide details regarding the delivery of various types of training:

- Existing employees training
- Initial CCSAS CSE application training
- Follow-up or refresher training after initial CCSAS CSE application training
- On-going training to reflect system application changes
- New employee training
-

The schedule will include the location of the training, the date and time of the training, the alliance trainer(s), and any CCSAS project or county staff required to participate in the training. Scheduling of the training will be disseminated to users via the internal secure web site.

In addition, the plan will describe the hardware and materials required for training, identify training sites, and provide a list of instructors and their respective skill levels. The plan also will include planning for the development of the following components:

- Software user manual instructions
- Quick reference guide instructions
- Tutorial instructions through e-learning
- Help text instructions through on-line help
- New desktop tools training, such as report writing tools
- Customization, i.e., tables

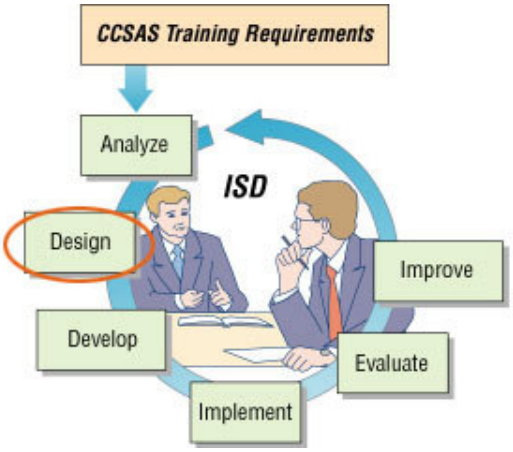
Finally, the CSE Version 2 User Training Plan will address the development of an assessment tool or proficiency test to measure training effectiveness, including how improvements are incorporated into the delivery of CCSAS CSE best of breed application training over time.

Understanding CCSAS CSE

In Phase II, the critical prerequisite to the design of training is for the alliance training team to understand the functional architecture of the new CCSAS CSE application. To accomplish this, the alliance training team's technical liaison will participate in design reviews conducted by the alliance development team. Based on the knowledge gained

in these meetings and review of subsequent design documents, our content developers, instructional designers, CCSAS CSE subject matter experts, and training technical architect will formulate concepts for the development of the e-learning courses. The tight integration of the alliance's training and application development teams is essential for the alliance training team to remain abreast of system functionality throughout the development cycle.

The alliance training team also will work closely with the alliance change management team to understand and incorporate the future business processes model as the context of the scenario-based learning activities.



Design

The next stage of the ISD methodology is **design.** In this section we address:

- Design principles of adult learning and scenario-based learning
- The design of the curriculum
- The design of an e-learning prototype

This stage involves using the assessment and training requirements data from the Analyze stage to execute a strategy for developing the instruction. During this stage, curriculum is designed to meet instructional goals determined during the Analyze stage. Our CCSAS CSE design approach is guided by two overarching design principles, as follows.

Design Principles

Design will be guided by two principles of curriculum development and delivery: 1) Adult Learning Principles, and 2) Scenario-Based Learning.

Adult Learning Principles

Adult Learning Principles recognize that adult learners:

- Learn effectively through problem solving and discussion
- Learn by doing
- Need to see a practical application for material learned
- Have different learning styles
- Need reinforcement and repetition
- Like to self-direct their learning if possible

The alliance's training solution, e-learning guided by a combination of on-site coaches and instructors in the classroom, is the ideal solution for busy child support enforcement staff. With this solution, the staff will learn what they need to know in the most effective and efficient way.

Scenario-Based Learning

In line with Adult Learning Principles, the alliance is developing scenario-based learning modules, providing trainees with reality-based experience using the CCSAS CSE system prior to cutover. For the CCSAS CSE project, scenario-based learning is preferable to presentation-only or rote learning methods because people learn best by doing. Figure 3C.13.3-5 presents the steps of scenario-based learning.

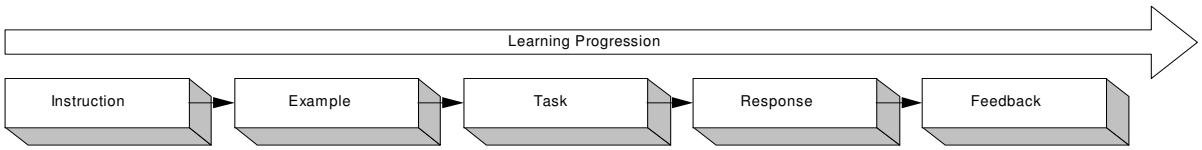


Figure 3C.13.3-5 Scenario-Based Learning – *A continuous training process where trainee is constantly involved in performing typical job functions.*

In scenario-based learning, the trainee is provided with instruction and examples to complete a case-processing scenario. Repetition and reinforcement are used so that the trainee masters the task at hand. The e-learning modules include several types of scenarios. Approximately fifty percent of the time, the workers will go into the CCSAS CSE training system and execute scenarios.

In scenario-based learning, the trainee is provided with enough instruction and examples to perform measured casework activity. The alliance training team will work closely with the change management team to understand and incorporate the new business processes models as the foundation of the scenario-based learning activities.

For example, they might actually update an absent parent's address, using data given to them in the e-learning module, while they are executing the "Case Update" module of the Case Management course. Additionally, the e-learning modules will use a simulated "show me, let me try" approach. The module will walk the trainee through a scenario, and then give them the chance to replicate it themselves.

Figure 3C.13.3.6 shows an example of an e-learning module in the midst of a "show me, let me try" scenario.

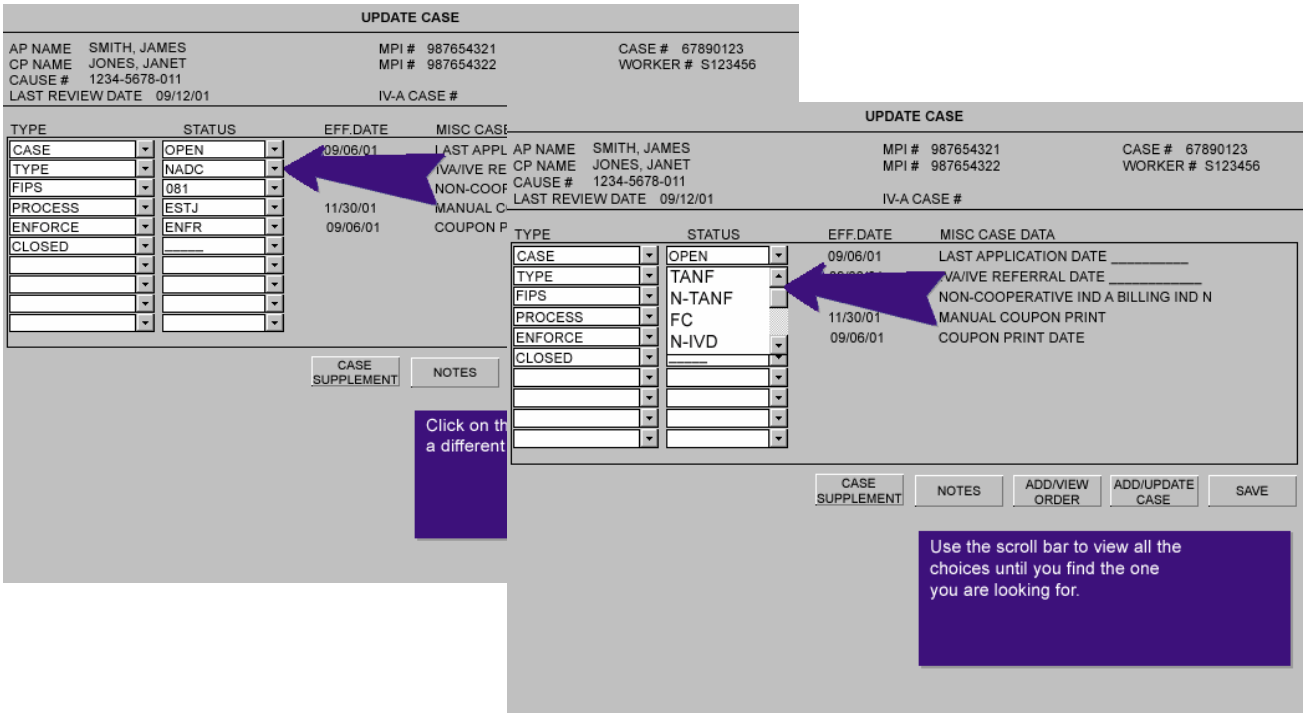


Figure 3C.13.3-6 'Show Me, Let Me Try' – *A technique used to involve the trainee in the learning process by 'showing' a particular skill and then expecting the trainee to 'try' the skill.*

Additionally, the scenario-based learning modules will be defined within the context of the CCSAS Business Model, which is the full set of new business processes to be implemented upon cutover to the new system. The alliance change management team conducts an effort to define the To-Be processes of the CCSAS Business Model. The alliance training team works closely with the alliance change management team to both understand the To-Be business processes as well as remain abreast of changes that may impact the development of scenario-based e-learning modules. The alliance training team will also work with the State personnel to identify and define the training scenarios to provide realistic scenarios.

Design Training Curriculum

The design of the training curriculum must take into account the job responsibilities of the user classes. We recognize that the classes and job responsibilities in child support enforcement offices vary from office to office and change over time. This will be particularly true as the offices undergo the change management process. We have therefore structured a flexible training curriculum that will allow the training liaison to choose appropriate training paths for their user classes, within the limits of our assumptions of the number of users who will participate in the training.

LCSA staff responsibilities vary from county to county based on the size of the county and its organizational structure. In some counties, workers are responsible for cradle-to-grave processing of a case and therefore will participate in the full range of training modules. In other counties, workers specialize in particular areas and therefore only need to participate in select training modules. The modular framework of the training curriculum allows LCSA training liaisons to select the most relevant training path for each LCSA user class based on scope of duties.

Development of the training curriculum begins with an understanding of both the CCSAS CSE functional architecture and the CCSAS Business Model process flows (To-Be process flows). E-learning training modules will be developed in line with the functional architecture of the new CCSAS CSE application and with child support worker activity, as described in the proposed courses for the LCSA staff listed in Figure 3C.13.3-7. The duration of the complete set of e-learning training modules will be eighty hours.

Probable Training Courses	**Probable Topics to be Included**
Overview	Overview, case structure, navigation, document generation, online help, workflow management, images, events, system use policies
Case Initiation and Update	Overview, case initiation, case update, case closure, scheduling
Establishment	Overview, paternity, genetic testing, court order demographics, guidelines/obligation calculation, health insurance, service of process
Enforcement	Overview, income withholding (earnings assignment order), tax and other intercepts, billing, credit reporting, passport denial, FIDM, license revocation, contempt's, liens, military, appeals, bankruptcy, adjustment after court order
Locate	Overview, automated locate, manual locate
Obligation Management	Overview, adding and modifying court order detail, review and adjust, overview of distribution and disbursement, interest calculation
Collections	Overview, receipting, payment identification,
Distribution	Overview, distribution, fees, adjustments, unreimbursed assistance, recoupments
Disbursement	Overview, disbursements, EFT, suspense
Common Services	Table maintenance and customization, archive and purge of historical data, appointment and hearing scheduling, event history, interstate, document generation
Support Services	Case work management, workflow management, document generation, report writing tools

Figure 3C.13.3-7 List of Probable CSE Version 2 Training Courses and Topics - *Modules and courses will be developed in line with the functional architecture of the new CCSAS CSE application and with child support worker activity.*

In addition to the LCSA staff, the same application training courses and materials will be delivered to the Help Desk staff.

Other CSE Version 2 specialized training topics will be delivered in a variety of ways. The System Administration materials will be delivered as part of the on-site support. Figure 3C.13.3-8 indicates the topic, method of delivery, and audience:

Topic	Includes	Audience	Delivery Mechanism
Program Monitoring	Reports generation and analysis	Up to 30 DCSS staff	Classroom training for ½ day
System Administration	Maintenance of user profiles, security, reference tables	Select staff	Materials and shadowing with on-site support staff
Customer Service	Customer Service Business Service of CCSAS CSE application	Select staff	2 hour e-learning module

Figure 3C.13.3-8 Topic, Method of Delivery and Audience – *Limited audiences will be required to take the specialized courses.*

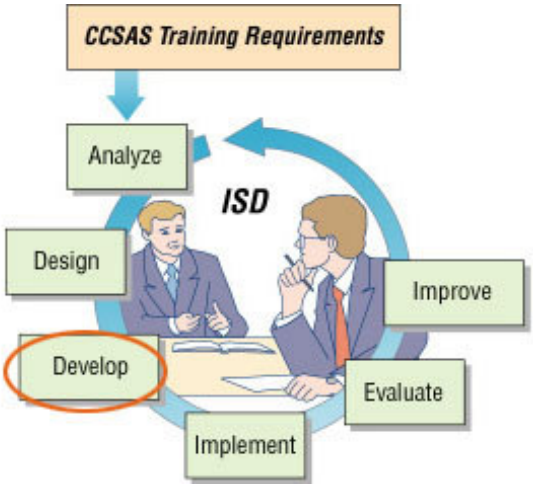
Design e-learning Prototype

In order for the CCSAS project staff to have a better idea of the final e-learning functionality early, the alliance team develops an e-learning prototype. We choose one module, such as genetic testing, and produce a high level, working version of the e-learning module. This prototype demonstrates the look and feel of the product, and includes an example of the different teaching tools. For example, even though the CCSAS CSE system will not be ready at this point, the prototype will show how the trainee receives instruction and performs exercises in the e-learning course**.**

Develop

The next stage is **develop**. In this section we address the development of the

- e-learning modules
- Learning Management System (LMS)
- Training system development
- Training reference materials



E-Learning Modules

Our CSE Version 2 training approach is built on an e-learning delivery foundation. For CCSAS, this approach provides several important benefits to California. First, it allows distribution of our training modules over the secured intranet site. It also provides

ongoing and continuous access to training even when no training sessions are scheduled. E-learning is especially applicable for training on the use of web-based applications to large, geographically dispersed audiences. As training must reach trainees in 58 California counties, the State's cost associated with travel and delivery has been one consideration in the selection of e-learning as a proposed delivery method.

In both the classroom and coach-assisted training, e-learning modules will be used to deliver scenario-based lesson content. In the classroom setting, the instructor will lead the trainees through each lesson via the e-learning solution, supplementing instruction by providing more specific contextual information and facilitating discussions between trainees. In the coach-assisted setting, trainees will access e-learning modules, perform exercises and complete the training. The trainee coach will be on site to answer questions and provide guidance.

In addition to the CCSAS CSE application curriculum, trainees will receive instruction on how to leverage the e-learning application and other user reference materials to support their work on the job. Once training is completed, trainees will continue to have access to the e-learning modules as both a source of reference as well as self-paced refresher training.

Learning Management System (LMS)

An LMS will be used to deliver CSE Version 2 e-learning to trainees through the internal secure web site, directly to the trainee's desktop at the regional training facility or at his or her office. Our proposed e-learning solution includes a learning management system used to manage and distribute course content, track attendance and monitor trainee performance on tests. The alliance considers an LMS key to meeting CSE Version 2 user training objectives. An LMS is especially beneficial when there are many learners, distributed over a wide geographic area, as in California. The LMS provides the following advantages for the CCSAS CSE project:

- Provides a mechanism for delivering e-learning modules over the secure intranet
- Allows course modules to be centrally managed. This makes it much easier to keep materials up-to-date and provides little opportunity for divergence of materials from a unified training approach
- Allows the scheduling, presentation, and management of multiple modules of online course content
- Efficiently and centrally manages trainee and instructor accounts
- Generates attendance sheets, tracks, and reports trainee course attendance

The alliance implements a robust, mature, and scaleable Learning Management System (LMS) for managing and delivering e-learning course content. The LMS features an intuitive web-based student-learning interface and a highly functional learning management utility for managing instructional materials, trainee accounts, and reports. The student-learning interface provides an easy way for the learner to interact with up to date training materials like task-based instruction, show me let me-try scenarios, and tests.

The learning management interface provides tools to manage courses and curriculum, enroll and track trainees, and monitor the overall effectiveness of the learning. Administrators can create profiles and assign a unique set of permissions and courses to these profiles. For instance, by assigning a group of users to a custom profile, a training manager can be confident that these users will be automatically enrolled in a curriculum unique to their needs.

Tracking learning is also a key element of the LMS. Because the LMS will be an AICC (Aviation Industry CBT Committee) compliant web-based computer managed instruction (CMI) system, it can track content that is designed according to the AICC recommendations. Examples of such trackable content are application simulation, Macromedia flash-based multimedia and even more traditional HTML tests. These tests can be scenario-based measuring the user's proficiency of using the application as well as multiple-choice, true/false and procedure oriented. All parameters that are tracked, including training effectiveness, can be analyzed visually as to the number of correct answers through the LMS report generator. Such reports can be securely viewed online, or sent as email to interested recipients. This provides training managers with documentation that they can use to prove the effectiveness of training.

E-learning Training Database Development

A training copy of the best of breed application and database is used for developing the scenario-based learning. The alliance will populate the e-learning database with selected, sanitized data from the conversion process for use with the scenario-based exercises included in the e-learning modules. The scenarios illustrate key points of the CCSAS CSE training and will be carefully version controlled in order to be in sync with the system status and the training presented at each training site. The e-learning database will be developed during Phase I and will become available during Phase II. It will have response times representative of the production system, and will be available from 7 am to 7 pm PST Monday through Friday.

The CCSAS CSE e-learning training database provides for:

- Presentation of cases and case situations that are keyed to illustrate specific learning objectives and that provide baseline knowledge of system concepts
- Individual case members, fragments of cases, and full cases to be used by trainees in exploring and expanding trainee experience with the system

Version control will be a key consideration in management of the e-learning training database. The appropriate version of the database must be loaded and/or transactions applied to create the version of data that support training scenarios to be delivered in a particular session. Additionally, as updates are made to the CCSAS CSE application, the e-learning training database will also reflect those changes so that the training environment remains consistent with the latest release of the CCSAS CSE application. Relevancy of a change to the CCSAS CSE application will determine when the change will be replicated in the training database and any affected e-learning modules. For those changes that visibly affect or impact users, the e-learning training database and/or e-learning modules should be updated at the conclusion of the current training session

when appropriate to prevent disruption to the training. Release notes will be made available for those users who have already attended a training session to make them aware of the update. For those changes that occur "behind the scenes", the training database and/or e-learning modules will be updated on a mutually agreed schedule.

Develop Classroom and Coach-assisted User Reference Materials

Our training approach combines both standard classroom training and coach-assisted training. Training reference materials are available to the trainee during training and the use of them is integrated into the course material. This allows trainees to know how to use the reference materials for ongoing assistance in answering "how do I" questions after cutover. These reference materials are used for both methods of training, the classroom and coach-assisted e-learning:

- On-Line Software User Manual Plan (SUMP) (CDL TM 085)

  The alliance will develop an online software user manual, which contains system information to assist child support personnel in the performance of their job duties. This online manual contains a search engine, glossary, index, and a substantial use of hyperlinks to increase its usefulness. It will be organized into logical modules based on system functionality. Development of this manual is largely driven by the development of the system functionality and by the availability of the corresponding design documentation. The alliance anticipates using RoboHelp as the development tool for the SUM. Users will have access during both e-learning and classroom training to the online Software User Manual.
  - In addition to the Software User Manual, the BP will develop an online Program Policy and Regulation (OPPs) database modeled after the SUM. OPPs' documents will not be integrated with the SUM; the OPPs database will be a separate entity. The OPPs database will be available to users when the Version 2 CSE system is available and will not be used during Pilot training. Materials provided by the State include:
    - DCSS Manual of Polices and Procedures (MPP)/Regulations)
    - Portions of the California Code of Regulations (CCR) pertinent to child support enforcement
    - Child Support Services (CSS) Letters
    - Child Support Services Informational Notices (CSSIN)
    - Local Child Support Services (LCSA) Letters
    - Family Support Division (FSD) Letters
    - Family Support Division Information Notice (FSDIN)
    - Policy Interpretation Letters (01/04)
  - RoboHelp will be used to develop the OPPs database, which will have the same look and feel as the SUM. It will include:
    - Table of contents – A list of hyperlinks to OPPs' documents arranged in categories and subcategories
    - Index – An alphabetical list of keywords and phrases, each of which hyperlinks to associated OPPs' documents
    - Search tool – A tool that allows users to search for words or combinations of words that occur in OPPs' documents

- The OPPs database will be accessed from within the CCSAS CSE system using a single button in the Global navigational bar. Clicking this button opens the OPPs database to a predefined default page.
- The OPPs database will not be context-sensitive. When users access the OPPs database, they will enter at a home screen and begin their search there.
- The OPPs database will not include a glossary, but will link to the DCSS Child Support Program Glossary. This feature requires internet access.
- State Provided OPPs' files from 1980 to April, 1999:
  - The State will provide to the BP a CD containing files developed using CA SupportBase - Folio Bound Views 3.11. This database uses the Folio Bound Views software called CABase which has the files in a text format that is needed for the OPPs database in RoboHelp. This is an older version last updated in the 1990's. The database will need to be upgraded to a current version before using an extraction toolset to extract the text files.
  - The BP to purchase the software needed to Upgrade Path to Folio 4.6, to migrate Folio InfoBase CA SupportBase files from Folio VIEWS 3.11 to the most current version.
  - Once the database is upgraded to a current version, the BP to purchase software to extract and export files into an electronic format that can be loaded to RoboHelp.
  - The BP will validate at this point that the extracted files match the original files in CA SupportBase – Folio Bound Views.
  - The State will review the extracted files to determine which still reflect current policy and will be added to the OPP database.
  - The State will be responsible for validating the current CA SupportBase "key" words associated to the files prior to loading into RoboHelp. This will be used by the BP to create the index.
  - The BP will load the extracted XML files into RoboHelp after BP and State validation is complete
- State Provided OPPs' files not contained in CASupport database
  - The State will provide the BP with all OPPs documents in a feasible electronic format, meeting all of the following conditions:
  - Each item in the OPPs database must be provided in a separate file, e.g. CSS Letters/Notices CSS 05-23 and CSS 05-24 would each be a separate file
  - It is recommended that the OPPs documents be provided in Microsoft Word (version 2000 or later) files with the ".doc" file extension. This will better the look and feel of the OPPs database. PDF files are acceptable, but the look will be that of a scanned page. The BP will not do any manual formatting of OPPs documents to improve the look.
  - It is recommended that OPPs content be in text format, not image format. Note: Scanned documents are typically in image format unless converted to text using Optical Character Recognition (OCR) software.
  - Files may not be encrypted
- The State will provide the "key" words to associate with the OPPs documents. This will be used by the BP to create the index.

- The State will provide the table of contents structure for the OPPs database
- The OPPs database will be updated as necessary, up to once per month prior to the Pilot Counties being implemented
- The OPPs database will be updated quarterly after pilot counties implementation
- The State will be responsible for providing completed PPR documents in the same format as during OPPs development: during the quarterly update process
- The BP will transition the OPPs database maintenance to the State during the Training transition period after Version 2 implementation

- E-learning System Data/Case Related Exercises

Because the CCSAS CSE application training will be child support activity based, there will be plenty of hands-on exercises of child support scenarios for the trainees to complete during training. In some instances such as financial training, trainees may be given a written sheet with scenario data. This will occur only if it is easier for the trainee to work from paper rather than the e-learning tool.

- On-Line Help

The CCSAS CSE application will include an on-line help function. This will support ready access to CCSAS CSE "how-to" information. The on-line help content will be derived from the SUM and will be developed utilizing RoboHelp.

- Quick Reference Guides

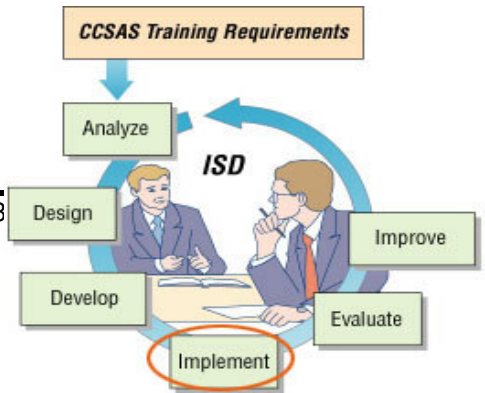
As the trainees move through the child support scenarios of case initiation, enforcement, locate, etc., in the both the classroom and coach-assisted training, we will provide reference materials to aid the users in working with the new system. Because child support activities are very process-oriented, users need bulleted lists of action steps presented in a visual way to aid them in learning these new multi-step processes.

- Frequently Asked Questions (FAQs)

Working with members of the on-site support, help desk and production support teams, the alliance training team will gather relevant FAQs for inclusion in an on-line FAQ tool.

Implement

The **implement** stage refers to the actual testing and delivery of the instruction through e-learning, both classroom-based and coach-assisted. It includes



- Testing course modules

- Revising
- Delivery
- Scheduling
- Regional training centers

Test Course Modules

A test plan will guide the consistent testing of courses as they are completed. This test plan will rely on a replicable procedure of evaluation for appropriate and correct content, presentation consistency, and technical execution. Testers, independent of the training content developers, will note problems for resolution. Additionally, courses will be tested in the first of the Training Preparation sessions. This first session will occur so that there is enough time to make necessary material adjustments based on feedback received from the trainees.

Iteratively Develop/Revise Course Content

Training curricula and supporting documents will be kept in synchronization with the application. Since course content will be managed from a centrally located LMS server, revisions will only need to occur in one place, eliminating the problem of duplication. A version control system will be put in place to manage this process. The alliance will include document version numbers, training material effective dates, and the corresponding software release version number on all training materials. Users already trained will be notified of revisions via email to each LCSA's training liaison.

Revision of course content will also be done in direct response to the testing stage described above, and will continue throughout Phase II, as appropriate. Updates will be distributed globally via the intranet, when CCSAS CSE system procedures change. A tight coordination between the training and development teams, managed through the proposed change control process, will help this occur. Depending on the relevancy of the updates – their impact on users – affected e-learning modules and the database will be updated on a mutually agreed schedule. For those updates that directly affect how a process or procedure is performed, the affected e-learning modules and database should be updated at the conclusion of the current training session when appropriate to prevent disruption to training conduct. All other updates will be performed on a mutually agreed schedule.

Delivery of Course Modules

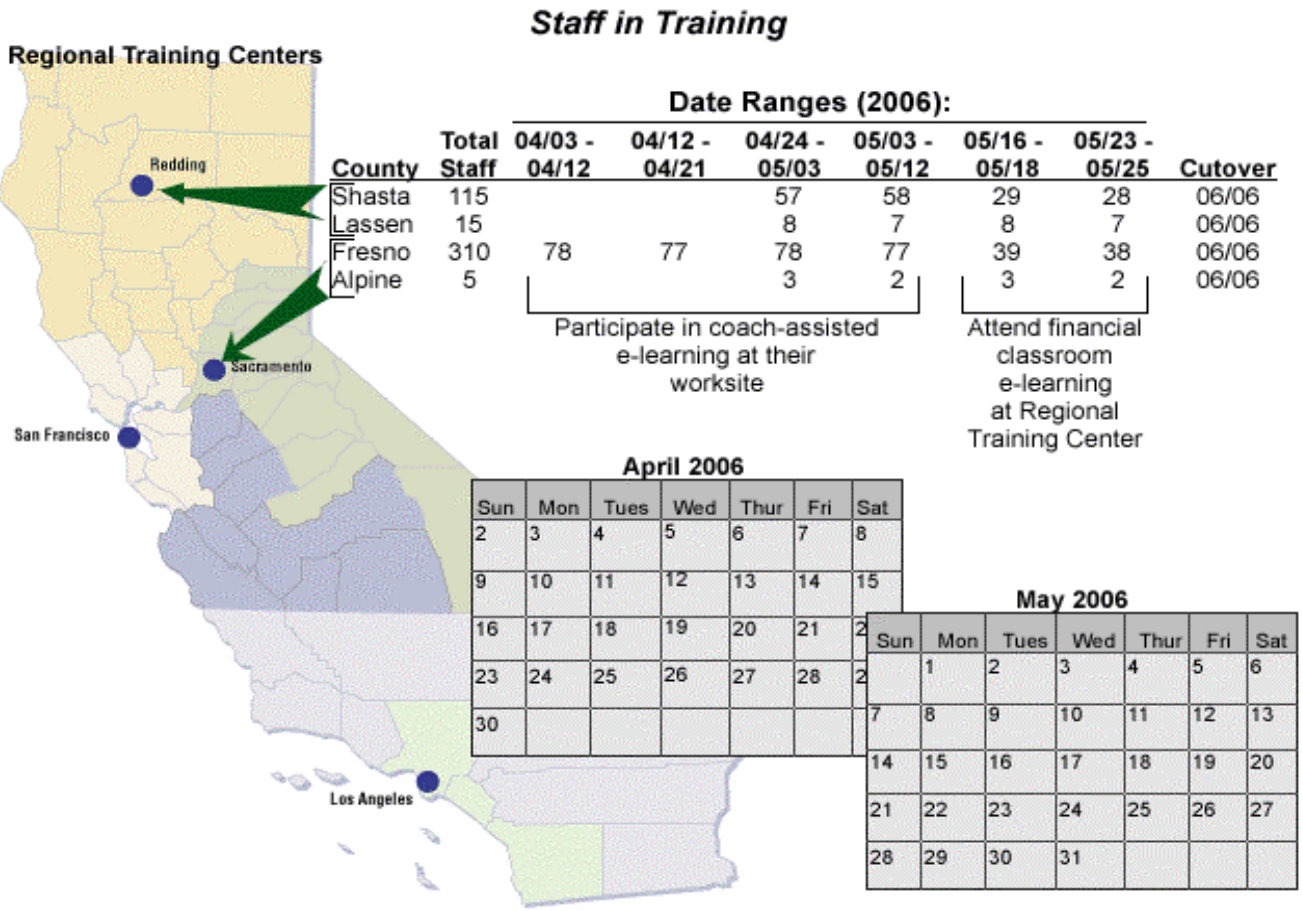
In the CCSAS CSE program, the relative complexity of system components varies across subject areas. The method used to deliver each module balances efficiency in delivery with learning requirements around complexity. Our proposed solution is crafted to achieve this balance by delivering the modules with the most complexity in an instructor-led classroom environment and the remaining modules through coach-assisted training.

The instructor-led format will be utilized for financial training. The remainder of the modules will be delivered in a self-paced format assisted by training coaches, who will

be available on site to resolve trainee questions on a one-on-one basis. This method is especially effective for staff that learns at different rates.
The e-learning modules are designed for eighty hours of learning, spread over ten days. Specifically, the case management courses should take a trainee seven and one-half days to complete, with coaches available for assistance. The financial modules are presented in the classroom over two and one half days. We have committed to a "just in time" approach for training, and have worked out a schedule to accommodate this. For counties with less than two hundred staff, the full training (coach-assisted and instructor-led) will occur in the five weeks prior to cutover. For counties with two hundred or more staff, training will occur in the eight weeks prior to cutover.

In order to achieve this schedule, we are assuming that fifty percent of staff is in training at any given time for those LCSAs with less than two hundred staff. For LCSAs with staff over two hundred, we are assuming that twenty-five percent of staff is in training at any given time. We are sensitive to the challenges associated with pulling staff from their normal duties, rearranging appointment and court schedules, etc. We commit to giving the LCSAs at least ninety days lead time so that they may adequately plan for the training interruption.

Figure 3C.13.3-9 presents an example of a schedule for 4 LCSAs of varying sizes.



County	Total Staff	04/03 - 04/12	04/12 - 04/21	04/24 - 05/03	05/03 - 05/12	05/16 - 05/18	05/23 - 05/25	Cutover
Shasta	115			57	58	29	28	06/06
Lassen	15			8	7	8	7	06/06
Fresno	310	78	77	78	77	39	38	06/06
Alpine	5			3	2	3	2	06/06

Sun	Mon	Tues	Wed	Thur	Fri	Sat
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	2
23	24	25	26	27	28	2
30						

Sun	Mon	Tues	Wed	Thur	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Figure 3C.13.3-9 Example Schedule – *Depicts 3 different scenarios for scheduling training, 2 counties with less than 50 staff, 1 county with more than 100 but less than 200 staff and 1 county with more than 200 staff.*

Figure 3C.13.3-10 presents the delivery method for each of the training modules.

Module	***Delivery Method***
• Overview includes Common Services and Support Services	• Coach-assisted training
• Case Initiation and Update	• Coach-assisted training
• Establishment	• Coach-assisted training
• Enforcement	• Coach-assisted training
• Locate	• Coach-assisted training
• Obligation Management	• Coach-assisted training
• Collections	• Classroom training
• Distribution	• Classroom training
• Disbursement	• Classroom training

Figure 3C.13.3-10 Course Delivery Methods – *All LCSA staff will receive at least some coach-assisted e-learning while a limited number will receive the classroom financial e-learning***.**

Train-the-Trainers

Selected CCSAS project and LCSA trainers and staff will be trained in eight sessions of thirty seats each. The sessions last up to 3 weeks (15 days), and will occur at the Sacramento regional training center. The curriculum will not only cover the features and functionality of the e-learning courses, but will provide basic training techniques and methods for both classroom and coach-assisted training, if time permits. Trainer guide materials will be provided to T4T attendees via the LMS. These sessions will address issues relative to the specific trainees attending and to the system in general, and offer guidance on frequently asked questions. LCSA staff who participate in these sessions will be considered "superusers," whom their peers may look to for CCSAS CSE answers or more formal training when coaches or on-site support staff are not available during the post-cutover period. The LCSA and DCSS trainers are also required to lend assistance in the coach-assisted phases of training. We feel that it will be of great benefit to those counties who choose to send either staff or trainers to the Train the Trainer sessions. Trainer Guides will be created as part of the development for the *Train the Trainers* course. Trainer Guides as scripted guides that will act as a road map for the instructors as they deliver classroom training. Step by step guides will promote consistency across sessions no matter which instructor delivers the curriculum.

Coach-Assisted E-Learning

Staff designated for coach-assisted training will be trained during the five or eight weeks prior to cutover, determined by the size of the LCSA as described above. While only a portion of designated staff receive financial training, all staff are scheduled for coach-

assisted training. Because of the location of the training (the staff person's worksite) and the e-learning application running in concert with the training database, this approach will give response times that are representative of production performance levels. Training will be scheduled so that a coach will be present during the training, providing a defined window in which the LCSA staff must complete the training.

To launch the e-learning activities at each site, coaches will meet with scheduled trainees and provide an overview of how the e-learning training works. The expected coach-to-trainee ratio will be 1-to-25. Once training is in progress, coaches will float on-site and be available to answer questions, demonstrate hands-on system functions, provide learning reinforcement, or help with scenario-based exercises.

Following the scheduled coach-assisted training time period, e-learning courses will still be available. This continued access to the courses via e-learning will allow workers to refresh their understanding of particular modules and complete additional modules for which they were not scheduled. Additionally, on-site support staff are present for the month of cutover and the month following cutover. If an LCSA has sent staff or trainers through *Train the Trainers*, these "superusers" are also a resource for their peers.

Classroom Training

We propose to present LCSA classroom training for the financial courses Collections, Distribution and Disbursement. The classrooms will accommodate thirty trainees, with two trainers per classroom. The trainees will be utilizing e-learning modules, with the same scenarios and work in the CCSAS CSE training database.

The BP will provide seats for financial management training to allow for 25% of LCSA staff to participate. The State is responsible for coordinating financial management training with DCSS and the LCSAs in the event that DCSS or a particular LCSA requests more than 25% of their staff to attend financial management training. These requests can be accommodated by the State in a variety of ways.

The classroom training will last two and one-half days per session, and will take place in the last two weeks prior to cutover. Trainees who receive classroom training will be able to register and take the e-learning courses from their work site for additional instruction, if and when they need it.

On-Site Support and Help Desk

Immediately following an LCSA's cutover to the CCSAS CSE application, users will have access to supplementary resources that can help them learn how to use the application. These include on-site support from alliance team members during the post-cutover period, and access to superusers and the Help Desk on an ongoing basis.

Scheduling

The training schedule will be driven by the implementation rollout schedule of the CCSAS CSE application in order to achieve the "just in time" training objective. Training will be delivered in two stages. The first stage is pilot in which three LCSAs are be

trained over a two-month period. The second stage is the statewide rollout in which the remaining 55 LCSAs and DCSS are trained over a 19-month period. Both classroom training and coach-assisted training will be conducted during these two stages.

The scheduling process will be conducted in cooperation with the LCSA training liaison. Based on the To-Be Model developed during the change management business process analysis, the LCSA's training liaison will determine which users need to attend which training modules. The results of this process will drive updates to the initial Pre-Assessment Survey results, reflecting changes in the number of users to be trained or worker functions that may have changed since the survey was first conducted.

As coordinated by supervisors of the staff to be trained and alliance Training Site Coordinator, a training schedule will be disseminated to each trainee, indicating when he or she is to be trained at his or her work site with the aid of training coaches and whether he or she will be attending the financial classroom e-learning. These notifications will be prepared in concert with appropriate supervisors of the users to be trained in order for the county and DCSS offices to coordinate any necessary travel arrangements for their staff.

Based on the number of 10,000 users, we have provided 10,000 training slots during CCSAS CSE rollout according to the number of users in each site and the rollout schedule proposed in Section 3C.13.4. If for some reason, staff is not able to attend during their scheduled training slot, the alliance will work with the LCSA training liaison to attempt to schedule that person in another slot. If no alternate training slots are compatible with the staff member, the LCSA will be responsible for providing training to that person through the LCSA's superuser and/or the ongoing availability of the e-learning modules.

Establish Regional Training Centers

Based on the proposed county rollout schedule, we estimate that four regional training sites will be required to train users during Phase II. Proposed sites to accommodate the geographic spread of users are Redding, Los Angeles, Sacramento, and San Francisco. We are open to discussing any alternative locations for regional centers that the CCSAS project might suggest, or discussing the availability of CCSAS project-provided centers. The planning of training sites for Phase II will begin 8-12 months prior to the time training begins in each individual site. When the geographic training locations have been finalized, the alliance training team will make final arrangements to secure facilities and prepare them appropriately for classroom training.

Figure 3C.13.3-11 Proposed Locations of Phase II Regional Training Centers – *Regional Training Centers are dispersed to decrease travel distance for trainees.*

Training centers require classrooms with 31 workstations and one printer. One scanner per site will also be available. Hardware and software will be tested during the readiness period to confirm that the necessary equipment is present at the training centers and functioning properly. The number of classrooms at each center is dependent on the size of the audience it is to serve.

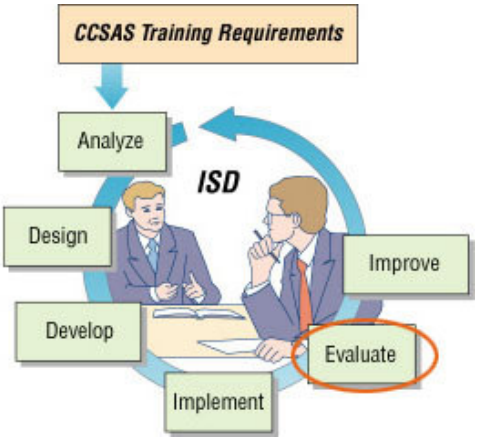
The regional training sites will maintain the e-learning environment with response times representative of production performance levels and be available from 7:00 am to 7:00 pm PST, Monday through Friday.

The alliance will close down the regional training sites located in San Francisco, Los Angeles and Redding in the month following the last course offerings at each of the sites.

Evaluate and Improve

The **evaluate** and **improve** stages measure the effectiveness and efficiency of the instruction. In this section, we discuss the:

- Evaluation feedback methods
- Attendance tracking



Training Evaluation: Efficacy Feedback Method

Training effectiveness will be tracked by two feedback methods, formative and summative.

Formative (as training is in progress)

Individual trainee performance in the scenario-based e-learning will be tracked by the LMS, stored as query data, and made available to trainers and managers in an automated web-based report format. For example, upon completion of a training module, trainees are asked to complete online assessments, which will be used to evaluate their understanding of the materials presented.

Criteria for evaluation of training effectiveness will tie directly to the goals and objectives determined as part of the ISD analysis. These criteria will be answered using the efficacy feedback methods described above. This data will be stored in the LMS, available to State and LCSA managers and project staff as on-demand reports. Based on the analysis of trainee responses to scenario-based exercises, trainers and instructional designers will identify places where training must be revised. The information gathered will drive required modifications to course modules and the training database.

Summative (upon completion of training)

Every trainee completes an online post-training effectiveness survey. Results will be tracked by the LMS, stored as query data, and made available to trainers and managers

in an automated web-based report format. At the end of each training course, the trainee is asked to complete an online survey that asks questions about the effectiveness of the instruction. The results of this survey are stored in the LMS and made available for analysis. Based on predefined security profiles, a manager or trainer who is interested in the effectiveness of a particular training course may log into the LMS and view the results of the survey.

Additionally, Post Implementation Review Reports (CDL TM 039) will be conducted for each implementation site. A portion of these reviews will be dedicated to the training program and will evaluate the effectiveness of the training over time.

Training Attendance (CDL TM 066)

The LMS will be used to track and report trainee attendance by unique trainee ID. Trainees will log into the LMS managed training environment using the unique ID, allowing the LMS system to track trainee attendance and performance automatically. Training Attendance Reports will be generated for both classroom and coach-assisted e-learning courses. These reports will reflect the course offered, date delivered, trainer's name, number of expected trainees, and number of trainees who attended.

3C.13.3.2.4 [RESERVE]

3C.13.3.2.5 Post-Implementation Training Program

The CSE Version 2 training program built during Phase II will continue to operate under alliance management for two years beyond implementation. At the end of that period, management control will be transitioned to the State . During this two year time period, the alliance will perform the following:

- System update training
- Management of one regional training center (Sacramento)
- LMS maintenance
- Training program transition

System Update Training

The e-learning courseware developed for CSE Version 2 training program during Phase II will be available throughout implementation and during the two-year post-implementation period. As modifications to the CCSAS CSE application are made, the courseware will be updated to reflect these changes in accordance to the level of effort described in the detailed task plan**.**

Any additional changes during this time will require evaluation and an estimate of the level of effort under an approved change order process. The impact these approved changes have on user training needs would be included in the assessment of the change order. We will work with the CCSAS project to determine what type of training effort best suits each change order. Based on the scope of these change orders, existing training courses and reference materials could be revised or new ones developed as appropriate.

LMS and Training System Maintenance

The Learning Management System will continue to be utilized for course registration and access to the e-learning courses. Its user database will require updates due to turnover, new assignments and new job requirements. The e-learning training database will be kept in synch with the CCSAS CSE application. Refreshing of data will be performed as needed.

3C.13.3.2.6 Training Program Transition

Training Transition Plan (CDL TM 063)

The alliance will prepare a Training Transition Plan (*CDL TM 063*). The Transition Plan will contain detailed arrangements for the alliance to transition the training program infrastructure and curriculum to the State, including course materials used during the training process. This plan will describe the activities that will occur during and at the conclusion of the two-year post implementation period and the timelines for the transition. It will also offer detail on the respective roles and responsibilities of the State and the alliance during this period. The Plan will be finalized later in the project incorporating lessons learned during training.

Transition Activities

The activities required for transitioning training to the State will be interwoven with the two-year post implementation activities. It is anticipated that State involvement with training activities will increase over this time period, with the heaviest involvement occurring during the final six months. The goal of transition is to prepare the State for the role of training program management and delivery. This includes an understanding and ownership of:

- The courses and modules developed for specific audiences
- The delivery timeframe requirements of each module
- The training delivery schedule and the processes in place for maintaining it
- LMS configuration and maintenance
- E-learning database configuration and maintenance
- Maintenance and update of existing e-learning courses
- *Train the Trainers* for State trainers

The alliance will look to the State to select knowledgeable and experienced staff to be transitioned over to these new roles. The following table presents estimates of the number of State staff our team will need to work with to complete the transition activity.

Transition Module	No. of DCSS staff	Method	Timeframe prior to end of project
Train the Trainer	Up to 10 staff	Classroom based Informal shadowing	24 months, with concentration the last 6 months
Learning Management system	2 staff	One week of classroom	6 months
E-learning Database Management	2 staff knowledgeable in database administration tasks	One week of classroom	6 months
E-Learning Module Maintenance	2-4 staff familiar with child support and instructional design, and proficient in tools such as Flash.	One week of classroom	6 months
Software User Manual and Online Help	2 staff familiar with the SUM and online help, and with RoboHelp.	One week of classroom	6 months

Figure 3C.13.3-12 Transition Activities – *Transitioning should be a continual process over the two-year post implementation period, but the majority of the formal transitioning will occur during scheduled periods.*

3C.13.3.2.7 Management

Alliance Training Team Organization

The alliance will organize our training staff into two primary teams. The team leaders will report to a single alliance Training Resource Group Lead. Additionally, a technical liaison will serve as an interface to the alliance development team. The two teams will be responsible for the following components of our training solution:

- Training Delivery – Responsible for classroom training and coach-assisted training. These responsibilities include training requirements analysis, training site coordination, scheduling of trainers and coaches, classroom setup and coach and instructor delivery.
- E-learning Development – Responsible for developing the e-learning training courses. This team will customize, configure, integrate, and manage the LMS used to schedule and monitor training, as well as to test trainees skills, knowledge, and proficiency. E-learning instructional designers will also work with classroom trainers to refine training strategies and methods of course delivery.

3C.13.3.3 Conclusion

User training strives to make users comfortable with the use of the new application through a training program that includes instruction via e-learning in coach-assisted and classroom formats. At the same time, the LCSA offices must be able to maintain daily operations with as little disruption as possible. The alliance believes that our training solution, including the transitioning of the training program to DCSS, will provide a training solution that meets the needs of the Child Support Program in California. The alliance training approach:

- Accommodates a variety of learning styles by combining training delivered by both instructors and training coaches
- Increases trainee retention by utilizing scenario-based learning techniques
- Provides users with consistent training with either delivery method
- Enables trainees to receive up-to-date information through electronic delivery of course materials
- Decreases disruption to the LCSA daily operations, since much of the training can occur at the workplace via coach-assisted training
- Provides a cost-effective solution to training the large number of users that need to be trained

3C.13.4 Hardware and Software Installation

The alliance has an established track record of successful, large-scale hardware and software installations. To support the CCSAS CSE implementation, we will draw on proven deployment strategies and methodologies to procure and deploy over 10,000 desktops and peripheral equipment to approximately 84 LCSA and state-level office locations throughout California. A well-timed and well-planned installation of hardware and software is one of the key components of the overall alliance implementation strategy.

3C.13.4.1 Overview

The alliance has assembled a partner team with the right blend of skills and experience to get the job done.

The alliance proposes a tailored and proven hardware and software installation strategy for the distinctive environment of CCSAS CSE. It offers substantial advantages to the CCSAS project team when measured against the critical success factors attached to a project of this size, scope, and complexity. In selectively drawing on the strengths of each partner, the alliance has created the right blend of skills, know-how, and experience to succeed in performing a very complex, exacting, and challenging hardware and software installation.

The IBM hardware and software methodologies we propose to use in the CCSAS CSE project, including the Network Deployment Methodology (NDM) and the Asset Management (AM) Methodology, have been validated in hundreds of installations. For example, in 1996 IBM installed 10,000 workstations, cabling, and 300 servers in 170 locations of the Child Welfare Services Case Management System (CWS/CMS) throughout California. The deployment spanned all 58 counties and provided Child Welfare Services offices with a common system. Service is ongoing and includes maintenance, tech-refreshes, and a single point of contact for technical support. Adhering to these refined methodologies will assist us to identify and reduce the challenges inherent in a deployment of the size and scope of a CCSAS CSE solution. They will serve us in meeting the overall hardware and software objectives throughout the installation process.

Objectives

Successful installations begin with a clear set of objectives. In the chart below, we offer an overview of those objectives with a summary of the alliance's approach to meeting each one.

Objectives	Approach
Reduce disruption to CCSAS end users during the hardware and software rollout process	We will: • Develop and conduct site surveys for each site to determine its hardware and software component needs • Perform installations outside of regular business hours when possible • Provide validation of installed hardware and software • Make resources and spare equipment available after installation • Plan and execute an effective transition to local technical support staff
Provide equipment and software based on the criteria for each type of asset	The alliance will perform a technology refresh to retire and replace old equipment
Install cabling, desktops configured with the standard software platform (described later), and other equipment, months in advance of production use	Through effective planning and scheduling, our approach yields adequate lead-time to address any problems that may be encountered during the installation process
Verify that provided hardware and software are installed and functioning correctly	We will conduct verification testing to confirm the functionality of hardware, desktop software
Execute an effective transition to end users and local technology support staff	We will provide an initial technical support team to ease the transition to the new hardware and software

Figure 3C.13.4-1 Hardware Objectives and Approaches. - *Our goal is to facilitate a smooth hardware and software installation while reducing disruption to the end-user community.*

In order to meet these objectives, we must understand and address a number challenges. *Figure 3C.13.4-2* illustrates the various challenges associated with the installation of hardware and software, including:

- A two-year hardware and software rollout timeline
- Geographically-dispersed sites
- The necessity to balance flexibility with uniformity
- The large number of users
- Deployment complexity

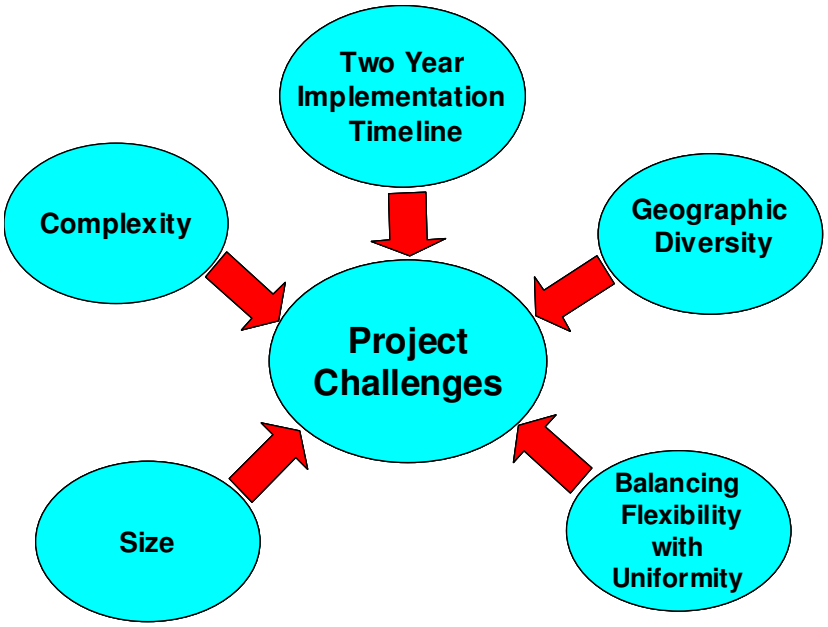


Figure 3C.13.4-2 Hardware and Software Project Challenges. - *CCSAS CSE introduces a number of distinct challenges.*

In the pages that follow, we will discuss the issues that touch on each objective, including:

- Our approach to accomplishing the Hardware/Software installation
- The key risks (and mitigation strategies) inherent in performing such a large-scale installation
- The assumptions we made in designing this installation process
-

3C.13.4.2 Our Approach

The alliance Hardware/ Software installation approach is designed to reduce disruption to LCSA end users and to mitigate risk by sequencing the plan in discrete segments, validating each step before moving on to the next.

The hardware/software installation will follow the overall CSE two-phased approach. Phase 1 will implement Version 1 of the CCSAS CSE application, the Statewide Services component, and transition 14 LCSAs to CASES. For Version 1, existing WAN, LAN and hardware will be used to access CASES, ARS and the Statewide Services component of the CCSAS CSE system. Phase 2 will implement Version 2 of the CCSAS CSE system, the “best of breed application”, in all LCSAs. During Phase 2, the complete rollout of new desktop, scanner, and printer technology will be accomplished to support local site end users.

3C.13.4.2.1 County Options

There will be three options for hardware / software rollout and support. These options were developed recognizing that there are varying degrees of infrastructure and support appropriate to the different needs of each county that will be incorporated in the CCSAS solution. The three options provide flexibility, while at the same time retaining a consistent model. Table 3C.13.4-8 identifies which counties have been assigned to which option.

The three options are summarized as follows:

Option 1 The Business Partner provides the equipment and support, including the desktop workstations, LAN, and administrative / problem support.

Option 2 The Business Partner provides the desktop workstations and LAN and administrative / problem support for the LAN infrastructure. The main difference between option 1 & option 2 is that option 2 counties are permitted to load additional software products to the desktops, and thus assume responsibility for Level 1 help desk and administrative and problem support for the desktop applications. The alliance retains responsibility for break fix support for the desktops and the LAN.

Option 3 The Business Partner provides the desktop workstations and break fix support for the desktops. The county provides the LAN, and all support except desktop break fix.

Each desktop regardless of the Option will contain the following CCSAS CSE standard software:

- Operating System (MS Windows XP)
- Browser (MS Internet Explorer)
- MS Office Suite
- Norton Anti-virus Software
- Adobe Acrobat Reader
- Tivoli Framework Manager Agent

Option 2 and 3 counties will be allowed to add additional software to their desktops. As a result these counties will be responsible for providing Level 1Help Desk problem identification and resolution services to their end users. Below is an outline of the different Options.

3C.13.4.2.1.1 Option 1

For Option 1 counties the alliance will provide the following assets and support:

Assets

- Desktop hardware with CCSAS CSE standard software
- LAN, including all cabling and switches
- MS Exchange mail service
- Image Server, Image Workstation(s), Scanners and Local Printers

Support:

- Desktop User Support, including standard applications and e-mail
- Desktop, LAN & email administration
- Desktop, LAN, Image Server, Image Workstation(s), Scanners and Local Printers break fix

Option 1 is designed for counties that have limited or non-existent hardware, software and technical support infrastructure. The alliance will provide and maintain these environments in a highly controlled fashion. Counties participating in Option 1 will not be allowed to load any additional software on the desktop, or any additional equipment (such as printers or file servers) to the LAN. As a result the alliance will provide all levels of Help Desk support to these counties. The Option 1 solution will also provide each user in the county with approximately 1 Gb of additional disk storage on the Image Server, available to store copies of MS Office Suite data files for backup purposes.

3C.13.4.2.1.2 Option 2

For Option 2 counties the alliance will provide the following assets and support:

Assets

- Desktop hardware with CCSAS CSE standard software platform plus Terminal Emulation
- LAN, including all cabling and switches
- Image Server, Image Workstations(s),Scanners and Local Printers

Support

- Desktop User Support for CSE Application Only (after County first contact)
- LAN infrastructure administration (after County first contact)
- Desktop, LAN, Image Server, Image Workstation(s), Scanners and Local Printers break fix (after County first contact)

For Option 2 the State / County will provide the following assets and support:

Assets

- Non-CSE applications and software
- E-mail system

Support

- Level 1 Help Desk first contact
- LAN user administration (e.g. passwords)
- User Support for all non-CSE applications
- Terminal emulation & e-mail

Option 2 is designed for counties that have some existing hardware, software and technical support infrastructure, and / or the need to provide their users with additional applications on the desktop such as emulator based access to legacy systems. The alliance will provide and install the LAN and the standard image desktops. The solution also provides for a default number of additional county-owned devices, such as existing printers or file / mail servers to be added to the new LAN by the alliance. The county will then be responsible for installing and configuring the additional software on the desktop, for administering the additional devices attached to the LAN, and for providing user support for these non-standard features.

3C.13.4.2.1.3 Option 3

For Option 3 the alliance will provide the following assets and support:

Assets

- Desktop hardware with CCSAS CSE standard software platform plus Terminal Emulation
- Image Server, Image Workstation(s), Scanners and Local Printers

Support

- Desktop User Support for CSE Application Only (after County first contact)
- Desktop, Image Server, Image Workstation(s), Scanners and Local Printers break fix (after County first contact)

For Option 3 the State / County will provide the following assets and support:

Assets

- Non-CSE applications and software
- E-mail system
- LAN, including all cabling, switches and firewalls
- Connection to other Networks

Support

- Level 1 Help Desk first contact
- User Support for all non-CSE applications
- Terminal emulation & e-mail
- LAN Administration & LAN break fix

Option 3 is designed for counties that have, and wish to retain, existing hardware, software and technical support infrastructure. The alliance will provide the standard desktops and attach them to county provided cable drops. The county is then responsible for installing and configuring the desktops with any additional software.

The following charts provide a side by side comparison of the assets and support of each of the 3 desktop options:

Assets	Option 1	Option 2	Option 3
Desktop Computer	alliance	alliance	alliance
Standard Software Platform	alliance	alliance	alliance
Email Service	alliance	LCSA	LCSA
Emulator Software	No	Y	Y
Additional Software	No	LCSA	LCSA
LAN Cabling & Hardware	alliance	alliance	LCSA
Image Server / Local Printers	alliance*	alliance	alliance

* The Image Server will also provide additional disk capacity to accommodate backup copies of user's non-CSE data files

Support	Option 1	Option 2	Option 3
First Contact Support	alliance	LCSA	LCSA
CSE Application Support	alliance	alliance	alliance
Desktop Administration	alliance	LCSA	LCSA
LAN Infrastructure Administration	alliance	alliance	LCSA
LAN User Administration	alliance	LCSA	LCSA
Email Service	alliance	LCSA	LCSA
Desktop Break Fix	alliance	alliance	alliance
LAN Break Fix	alliance	alliance	LCSA
Image Server / Local Printers	alliance	alliance	alliance
Software Distribution	Push	Pull	Pull

Each county will be implemented fully in one and only one desktop option. These desktop options will be applied consistently among counties. Option 2 and 3 counties will be responsible for converting any files from their existing desktops to the new CCSAS supplied desktops.

Software distribution for desktop software fixes will be managed centrally. For option 1 sites, software will be distributed to the Image servers in each county, which will also function as software distribution servers. The desktops will be updated using a push approach from the county Image server. For option 2 & 3 sites, the county staff will be responsible for updates by pulling software from the alliance provided central server. The Norton anti-virus software has the capability to be scheduled to automatically seek updated virus signature files via the Internet from the publisher's web site. Adobe Acrobat updates are also available via the Internet, and these updates will be made available in a coordinated manner to users according to the project's needs.

Desktops will be deployed to four Regional Training Centers in:

- Redding in Shasta County
- Los Angeles
- Sacramento
- San Francisco

These will follow the Option 1 model.

Desktops deployed to the CSE Project Office and Customer Service Support Centers will follow the option 1 model.

Desktops deployed to the DCSS office, CCR will follow the Option 3 model.

The remainder of this *Approach* subsection is divided into the following three subsections where the alliance's approach to the hardware and software rollout is more fully described:

- **Strategy** – The steps and approach the alliance will use to rollout the hardware and software.
- **Proven Methodologies** – Our NDM and AM methodologies are the best practices that govern our hardware and software strategy, including preparation, coordination, verification, and recovery activities
- **Equipment and Software Specifications** – Estimated CCSAS CSE hardware and software installation schedule and specifications

3C.13.4.2.2 Strategy

The proposed hardware and software installation process, from beginning to end, is illustrated in *Figure 3C.13.4-3*.

HW/SW Installation Process

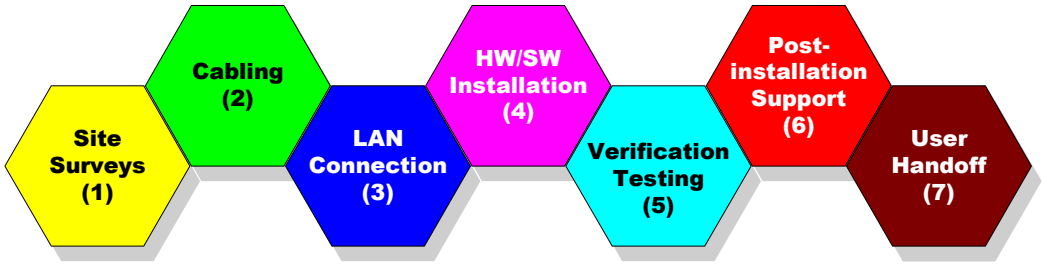


Figure 3C.13.4-3 HW/SW Installation Process - *The process of hardware and software rollout begins with site surveys, followed by installation, verification, and, eventually, handoff.*

The hardware and software installation process requires a timed series of seven steps that will be conducted for each site. The details and extent of the process for each site shall depend on the desktop option selected for the site.

3C.13.4.2.2.1 Site Surveys

CCSAS CSE Version 2 Application Rollout
Plan and prepare for hardware and software
Develop and verify Site and Readiness surveys
Conduct site-cabling surveys

A site survey will be administered in each LCSA and State-level site in order to develop the requirements for each one's distinctive hardware and software configuration. The surveys will provide information to be used for:

- Cabling plans
- Appropriate housing for local area network components (hubs / switches / firewalls)
- Locations for Image Servers, Image Workstation(s), Desktop workstations, Scanners, and Local Printers

The resulting information will be used to help validate the Existing System and Facilities Assessment and to revise the site-specific components of the overall Hardware and Software Installation Plan (CDL TM 067).

3C.13.4.2.2.2 Cabling

CCSAS CSE Version 2 Application Rollout
Install new cabling for option 1 & 2 sites
Provide necessary input to the Change Management Resource Lead for the development of the outreach plan for CCSAS CSE application clients and external entities
Procure hardware and software

The cabling step involves the installation of cable for option 1 and 2 counties for desktop equipment in the Local Area Network (LAN). Cabling includes the use of standard cable (Category 5), segment planning, redundant looping where appropriate, and end-to-end signal testing to validate that each drop is correct. We propose cabling in a compressed six-month period, beginning seven months prior to the first installation of the Version 2 application. . This allows adequate lead-time to address issues that may delay the installation of cabling, e.g., unanticipated office moves, the discovery of asbestos, or constraints on those facilities designated as historical landmarks. Additionally, this approach reduces the potential periods of down time that may otherwise result between installations. Cabling completion deliverables will include appropriately wired and labeled patch panels and site cable diagrams.

3C.13.4.2.2.3. LAN Connection

CCSAS CSE Version 2 Application Rollout
Coordinate with LAN designers and installers
Establish LAN connection at each option 1 and 2 site. Connect to WAN routers and test connection to router.

Connectivity and network services will be provided through the Local Area Network (LAN). We will install network switches and hubs for option 1 & 2 sites (as determined by the statewide baseline) and use laptop-based test equipment to confirm those appropriate protocols were sent and received.
The CCSAS CSE solution requires no local File or Print Services. Since the application will be Web-based, the system, reference, and data files will be managed centrally. Printers will be controlled via the Wide Area Network (WAN); print spooling will also be managed centrally. The CCSAS CSE solution does not include LAN support servers, with the exception of image servers that will be centrally administered. For option 1 counties, the image servers shall also serve as file servers for user backup of their desktop files.

Option 2 counties may choose to have their own LAN servers connected to the CCSAS CSE network and provide their local services via a single networking solution. Specific site needs will be discussed and documented as part of the site-specific Hardware and Software Installation Plan (SOW 13.4.1 / CDL TM 067) sections.

3C.13.4.2.2.4 Hardware and Software Installation

CCSAS CSE Version 2 Application Rollout
Install hardware and software with desktop installations to begin (on average), three months before a location is scheduled for conversion to CSE Version 2

The hardware/software installation step involves the deployment of desktop workstations, printers, and other peripherals at the site.

The desktop workstation installation will be performed on a Just-In-Time (JIT) basis, where installation will be accomplished for each site three months prior to cutover. This will allow ample lead-time for both user familiarization and the CCSAS CSE application training prior to cutover. It will also extend the productive life of the new equipment and delay the start of the equipment replacement clock.

Whenever possible, hardware and software installations will be performed outside of regular business hours to limit disruption at each site. The installation process will be performed in two steps:

(1) Equipment installation
(2) Technical training on the use and maintenance of scanners and printers equipment

To limit site disruption, spare hardware and technicians will be available to quickly remedy equipment malfunctions that may occur during the installation process.

No de-install services for existing equipment are provided in the scope of this contract.

If the CCSAS project team removes hardware, an inventory of equipment that was removed to storage will be required. In the unlikely event of a system failure, the de-installed desktops will then be available to recover the old system.

3C.13.4.2.2.5 Verification Testing

CCSAS CSE Version 2 Application Rollout
Execute tests of desktops and peripherals for all sites
Perform walk-through of installed equipment to verify proper transport, network, peripheral, and software activities, performance, and functionality for all sites

During verification testing, desktop workstations and peripherals will be tested to verify that they operate and properly communicate. The final site verification will be performed via a “walk-through” to confirm the installation, to verify proper transport, network, peripheral, and software activities, performance, and functionality. Upon verification, equipment will be turned over to workers for their use. This installation process will reduce transition disruption by verifying that the hardware and software was properly installed and functioning prior to end-user participation.

3C.13.4.2.2.6 Post-installation Support

CCSAS CSE Version 2 Application Rollout
The site-based technical support personnel will take over responsibility for technical issues (as appropriate for the county option).
Assistance will be available through the alliance for the first three days of operation to answer questions and to reinforce training in response to calls for service

At this stage, the equipment will be operating properly and the site will be in a "maintenance mode." The site-based technical support personnel and Local County Help Desks (for options 2 & 3) will take over responsibility for technical issues in coordination with the CCSAS CSE Help Desk. On-site assistance will be available through the alliance for the first three days of operation to answer questions and to reinforce training in response to calls for service. Such post-installation support will further ease the transition to the new hardware and software.

3C.13.4.2.2.7 User hand-off

CCSAS CSE Version 2 Application Rollout
Provide routine user training on hardware and software components
Provide installed software and hardware to end users for use
Provide onsite presence to answer user questions, resolve post-installation issues
Initiate hardware, software and helpdesk support

In this final step, the alliance will provide training in routine equipment usage, such as printer toner cartridge replacement, paper jam management, etc. After this usage training, the State/county representative(s) will accept and sign off the installation. After the hand-off is completed, maintenance and support issues for the hardware, operating system, and Web browser will be reported and will be either handled remotely or by local site technical support personnel. A toll-free technical support hotline, will be provided in the event of unexpected component failure.

3C.13.4.2.2.8 Hardware Refresh

The duration of the CSE contract extends beyond the life of the initially deployed equipment. For that reason, the alliance installed hardware shall be refreshed (commonly referred to as a tech-refresh), on a scheduled basis
The alliance will act as the custodial owner of hardware and software assets until transfer is made to the CCSAS project team. Given the extended timeline of this project, tech-refresh cycles will occur prior to asset transfer. The tech-refresh results in the State CCSAS project team receiving and assuming ownership of assets that are up-to-date, in accordance with the timing criteria selected for retention, upgrading, and replacement.

3C.13.4.2.2.9 Software Refresh

Application software will be Web-based. All the Web servers supporting this software will be centrally controlled. Such central control will permit application versions to be applied to all users simultaneously by refreshing the central Web server(s) with the latest copy.

Each time the user accesses the CCSAS CSE application, he or she will be accessing the most current software release.

Each software release will demonstrate to the CCSAS project that the application software can be regenerated.

The Release Notes (SOW 13.4.4 CDL TM 070) and Software Version Description (SVD) (SOW 13.4.2 CDL TM 068) will be used to communicate changes on the CCSAS CSE application to affected staff.

Upgrades to non-application software that is resident on the desktop and that is an integral part of the CCSAS CSE solution will be tested for use prior to deployment in the field. That software will include the desktop operating system and the desktop Internet browser. Option 1 counties will not be permitted to update or alter the software loaded to their desktop workstations. Option 2 & 3 counties may update or alter the software, but will be responsible for the support and administration of their desktops.

3C.13.4.2.2.10 Recap Hardware and Software Installation Activities

The alliance hardware and software approach allows adequate time to deal with potential obstacles and gain more active use for desktops before they are replaced in accordance with a three-year technology refresh cycle. The following table serves as an easy reference and review for the activities that will take place in each of the high-level hardware and software installation processes previously described. The installation activities in the tables below are listed in order of relationship, *not necessarily in order of execution*.

Phase II – Best-of-Breed CCSAS CSE Application Rollout
1. Site Surveys
Plan and prepare for hardware and software
Conduct site-cabling surveys
Develop and verify Site and Readiness surveys
2. Cabling
Install new cabling for option 1 and 2 sites
Provide necessary input to the Change Management Resource Lead for the development of the outreach plan for CCSAS CSE application clients and external entities
Procure hardware and software
3. LAN Connection
Coordinate with LAN designers and installers
Establish LAN connection at each option 1 and 2 site. Connect to WAN routers and test connections
4. HW/SW Installation
Install hardware and software with desktop installations to begin (on average), three months before a location is scheduled for conversion to CSE Version 2.

5. Verification Testing
Execute tests of desktops and peripherals for all sites
Perform walk-through of installed equipment to verify proper transport, network, peripheral, and software activities, performance, and functionality for all sites
6 Post-installation Support
The site-based technical support personnel will take over responsibility for technical issues (as appropriate for the county option)
Assistance will be available through the alliance for the first three days of operation to answer questions and to reinforce training in response to calls for service
7. User Hand-Off
Provide routine user training on hardware and software components
Provide installed software and hardware to end users for use
Provide onsite presence to answer user questions, resolve post-installation issues
Initiate hardware, software and helpdesk support

Figure 3C.13.4-4 Hardware and Software Installation. *This table depicts the hardware and software installation events in order of relationship, not in order of execution***.**

3C.13.4.2.2.11 Recovery of Business Systems

In the unlikely event of an aborted installation due to a hardware failure, software failure, or other problem, contingency procedures are in place to facilitate the recovery of business systems and processes. Given that the hardware and software is installed several months *prior* to cutover, it is unlikely that a problem of this nature may result in a considerable impact on business operations at the time of installation. In the event that a problem does arise and the installation of new hardware and software is aborted, the hardware and software belonging to the original consortia system is left in place until the problem(s) is corrected and installation is resumed.

Problems encountered during or immediately following, cutover are more complex. An aborted cutover requires recovery of the LCSAs original business systems and procedures, which are placed back into service until the problem is corrected. The Business Process Transition Plan (CDL TM 056) will address the business rollback plan. In coordination with this plan, the Implementation Contingency Plan (CDL TM 071) addresses recovery of the consortia's physical hardware and software.

3C.13.4.2.3 Proven Methodologies

The alliance uses IBM's proven methodologies to promote the successful deployment and management of the hardware and software installation process. The Hardware and Software Installation Plan (SOW 13.4.1 / CDL TM 067) will be developed according to the following standards and procedures in these methodologies:

- **Network Deployment Methodology (NDM)** determines rollout requirements and provides for how to manage risk, procure equipment, fulfill contract provisions, and develop schedules and work plans.
- **Asset Management (AM) Methodology** provides the tracking capability needed to support activities, which include hardware and software maintenance, tech-refresh, and the moves, adds, and changes that occur in a dynamic IT environment.

3C.13.4.2.3.1 Network Deployment Methodology (NDM)

The alliance brings to the CCSAS effort IBM's proven Network Deployment Methodology (NDM), which has been used extensively in large-scale public sector deployments. The NDM is an integrated system of evolved tools, best practices, and road-tested processes that have been successfully used in many large network deployments.

Network Deployment Methodology is a systematic process that determines key processes, people and tools.

As noted above, NDM is a systematic process used to determine rollout requirements, manage risk, procure equipment, fulfill contract provisions, and develop schedules and work plans. The NDM methodology integrates detailed processes and procedures with a governance and communication plan, staffed by the right resources, and supported by the right tools.

NDM packages an array of thoroughly tested tools specifically designed for technology rollouts and is well suited to the specific challenges of a large-scale, geographically diverse hardware/software installation. It is a direct result of more than two decades of network management and deployment experience.

NDM provides an end-to-end solution with a single source for hardware and software installation needs. The full range of customer services include:

- Strategy and standards definition
- Planning and implementation
- Ordering support/procurement
- Logistics
- Pre-delivery preparation
- Configuration, installation, and testing
- User training
- Asset control
- Hardware maintenance

3C.13.4.2.3.2 Asset Management (AM) Methodology

As stated earlier, the deployment of over 10,000 desktops and other equipment is the beginning of a large undertaking of considerable financial, managerial, and operational importance. In fact, the process of asset management begins *before* deployment with the selection of desktops and their acquisition/procurement.

Asset Management (AM) methodologies provide the means to support procurement, equipment deployment, maintenance, technology refresh, and other activities related to the life cycle of depreciable assets.

Asset management is a response to the life cycle of depreciable assets, beginning with planning for acquisition, followed by procurement, assignment, installation, configuration, moves, additions, and changes, periodic tech-refreshes, and finally ending in retirement or disposal.

AM is the process for capturing, integrating, and maintaining technical and financial information about IT assets from *requisition* through *disposal*. AM enables the alliance to make informed decisions and effectively manage operations in the following areas:

- Strategic Planning and Budgeting
- Deployment
- Contracts Management (Software, Maintenance, and Lease)
- Service Request Management (Move, Add, Change)
- Problem and Change Management
- Configuration and Demographic Tracking
- Process compliance, Service Level Agreements, and Approvals

Figure 3C.13.4-6 Asset Management Methodology - *The Asset Management Methodology covers the multiple aspects of IT assets, from planning to deployment to management.*

Reduced cost of ownership, improved productivity, ease of migrating to new technologies, and extending the useful life of hardware and software assets are among the many benefits of a robust asset management program.

The AM methodology provides the critical decision-making support to carry out these operations and realize key benefits for the CCSAS CSE end users. Overall, the Asset Management Methodology:

- Reduces the Total Cost of Ownership (TCO)
- Facilitates standardization efforts
- Improves IT budget management and planning
- Improves management of a distributed computing environment
- Improves productivity
- Assists with the migration to new technology
- Extends useful asset life
- Portrays accurately the value of assets in internal and regulatory reports

Tracking is a crucial component of asset management because of the complexity inherent in capturing changes that occur in a dynamic IT environment between *acquisition* and *disposal*. It entails the ability to assign and track the location and ownership of assets through adds, changes, deletes, and reassignments of hardware and software.

The AM methodology supports:

- Developing a Managed Elements List for tracking
- Establishing a centralized receiving process for asset control, including tools for warehousing, inventory, and receiving activities
- Establishing procedures to update the asset repository
- Management IT asset inventory
- Reconciling of variances or discrepancies in inventory records
- Auditing IT assets – identifying and resolving variances and exceptions
- Updating license information – renewal and vender invoice management
- Managing end-of-hardware life cycle – replacement notification
- Preparing of assets for disposal at the end of hardware's life cycle
- Administering Asset disposal process – refurbish, donate, scrap, or sell and data base update
- Managing Service Level Agreements (SLA) – managing in order to meet the agreed-upon service levels

To support tech-refreshes for CCSAS CSE, AM will generate reports to identify assets that should be refreshed based on the criteria selected for each type of asset. The reporting system will then generate a Procurement Request or a Change Request. The Procurement Request will begin the process *renewing* the license for software or *replacing* the hardware and software for the end-of-life-cycle event. Finally, a Change Request will be initiated to upgrade, remove, or replace the existing asset.

3C.13.4.3 Moves, Adds, and Changes

Once installed it may be necessary for equipment to be moved within an individual office building or between office buildings. When this need occurs in any county the county technical staff or single point of contact for facilities will contact the alliance Help Desk. For Option 3 counties the moves will be performed jointly by alliance and county technical staff.

INTRAOFFICE MOVES ASSUMPTIONS:

1. Moves must be requested and scheduled through the Help Desk 30 days prior to the work being performed.
2. Moves are limited to 1,500 per year.
3. Construction, cabling/wiring, circuits, and equipment have not been factored into the estimates.
4. Cabling to the site wiring closet already exists.
5. Moves, adds, and changes will not begin until all LCSAs are rolled out
6. To qualify for an intra-office move, the site must have a minimum of 6 machines per visit.
7. Moves do not cover adjustments or additions to network configurations (i.e. IP addresses).

OFFICE MOVES ASSUMPTIONS:

1. Moves must be requested and scheduled through the Help Desk 45 days prior to the work being performed.
2. Office moves are limited to 5 per year.
3. Office moves will not begin until all LCSAs are rolled out.
4. One office move is assumed for year 5.
5. Five office moves have been assumed for year 6.
6. Four office moves have been assumed for year 7.
7. To qualify for an office move, the site must have a minimum of 125 workstations, one server, seven printers, one scanner, one router, and three switches moved per visit.
8. Construction, circuits, and cabling/wiring have not been factored into the estimates.
9. There will be no site readiness inspection.
10. The site contact must provide the Contractor with the appropriate location and information for the activities.
11. Moves do not cover adjustments or additions to network configurations (i.e. IP addresses).

If the State requires support in excess of or contrary to these estimates, these would be submitted as change requests and subject to Change Request Management.

3C.13.4.4 Equipment and Software Specifications

The final specifications for workstations, network equipment, and cabling will be determined during creation of the Hardware and Software Installation Plan (CDL TM

067). The updated specifications will accommodate technological advances for the hardware and software products the alliance proposed but will keep costs within the proposed budget for that equipment. *Figure 3C.13.4-7* contains the specifications for the cabling, hardware, and software that has been identified for rollout to the LCSA and State-level sites:

Note: Equipment and Software Specifications are estimates based on the current environment.

Equipment and Software Specifications
Cabling -- Category 5.
Desktop Hardware
IBM NetVista M Series Desktop
Pentium IV 1.6 GHZ Processor
256 megabytes of RAM
40 gigabyte Hard Drive
48X CD-ROM
10/100 Ethernet Card
Warranty three years parts / 1 year labor
17-inch Monitor, .27-dot pitch
Scanning, Printing, and Server Hardware
Fujitsu 4099 Scanners
Fujitsu 4097 Scanners
IBM xSeries 342 Image Servers
IBM Infoprint 1125 Printers
Software
MS Windows XP Operating System
Microsoft Internet Explorer version 5.x

Figure 3C.13.4-7 Proposed Equipment and Software Specifications - *Cabling, hardware, and software have been carefully chosen to meet the CCSA project's requirements.*

Figure 3C.13.4-8 identifies the total number of devices that will be rolled out as part of the CCSAS CSE effort along with anticipated tech-refresh requirements. The quantities and distributions described in this table have been provided for budget and planning purposes only. The actual distribution of hardware at each of the sites will be determined after TM 038 Pre-Implementation Readiness Assessment has been completed. Deviations in total equipment quantities will be handled through the Project Scope Change Management process.

	Hardware Quantities										
Site/(Desktop Option)	**Desktop**	**Printer**	**Scanner**	**Option**	**Switch**	**Image Server**	**ImageWS**	**Cutover Month**	**Install Month**	**3 Year Refresh Month**	**5 Year Refresh Month**
Refresh (Years)	**3**	**3**	**3**		**5**	**5**	**3**				
Alameda(3)	244	12	2	3		1	3	55	51	0	0
Alpine(1)	3	2	1	1	2	1	1	47	43	79	0
Amador(1)	15	2	1	1	2	1	1	52	48	84	0
Butte(2)	141	7	1	2	2	1	1	51	47	83	0
Calaveras(1)	13	2	1	1	2	1	1	43	39	75	0
Colusa(1)	9	2	1	1	2	1	1	58	54	0	0
Contra Costa(3)	248	12	1	3		1	2	51	47	83	0
Del Norte(1)	30	2	1	1	2	1	1	52	48	84	0
El Dorado(2)	66	3	1	2	4	1	1	46	42	78	0
Fresno(3)	308	15	3	3		1	4	47	43	79	0
Glenn(1)	13	2	1	1	2	1	1	58	54	0	0
Humboldt(1)	78	4	1	1	6	1	1	50	46	82	0
Imperial(1)	39	2	1	1	2	1	2	59	55	0	0
Inyo(1)	8	2	1	1	2	1	1	55	51	0	0
Kern (2)	253	13	2	2	10	1	3	59	55	0	0
Kings(2)	73	4	1	2	2	1	2	44	40	76	0
Lake(1)	35	2	1	1	2	1	1	54	50	86	0
Lassen (1)	13	2	1	1	2	1	1	47	43	79	0
Los Angeles(3)	1,693	85	5	3		1	15	62	58	0	0
Madera(1)	46	2	1	1	2	1	1	45	41	77	0
Marin(2)	47	2	1	2	2	1	1	57	53	0	0
Mariposa(1)	9	2	1	1	2	1	1	48	44	80	0
Mendocino(2)	41	2	1	2	2	1	1	55	51	0	0
Merced(1)	123	6	1	1	2	1	2	45	41	77	0
Modoc (1)	8	2	1	1	2	1	1	53	49	85	0
Mono(1)	5	2	1	1	2	1	1	46	42	78	0
Monterey(1)	131	7	1	1	2	1	2	43	39	75	0
Napa(2)	42	2	1	2	2	1	1	54	50	86	0
Nevada (1)	47	2	1	1	2	1	1	53	49	85	0
Orange(3)	713	36	2	3		1	3	48	44	80	0
Placer (2)	87	4	1	2	4	1	1	48	44	80	0
Plumas(1)	13	2	1	1	2	1	1	51	47	83	0
Riverside(2)	474	24	3	2	12	1	5	58	54	0	0
Sacramento(3)	276	14	1	3		1	3	44	40	76	0
San Benito(1)	24	2	1	1	2	1	1	43	39	75	0
San Bernardino(3)	458	23	3	3		1	4	52	48	84	0
San Diego(3)	700	35	3	3		1	6	57	53	0	0
San Francisco(3)	138	7	1	3		1	3	60	56	0	0
San Joaquin (2)	200	10	1	2	2	1	2	53	49	85	0
San Luis Obispo(3)	56	3	1	3		1	2	39	35	71	0
San Mateo(3)	116	6	1	3		1	2	59	55	0	0
Santa Barbara(3)	122	6	1	3		1	2	45	41	77	0
Santa Clara(3)	337	17	2	3		1	3	54	50	86	0
Santa Cruz(2)	70	4	1	2	8	1	1	44	40	76	0
Shasta (1)	115	6	1	1	10	1	2	47	43	79	0
Sierra (1)	2	2	1	1	4	1	1	50	46	82	0
Siskiyou (1)	45	2	1	1	2	1	1	45	41	77	0

	Hardware Quantities										
Solano(2)	145	7	1	2	6	1	2	46	42	78	0
Sonoma (2)	165	8	1	2	10	1	2	40	36	72	0
Stanislaus(2)	197	10	1	2	6	1	2	53	49	85	0
Sutter(1)	46	2	1	1	2	1	1	40	36	72	0
Tehama(1)	32	2	1	1	2	1	1	51	47	83	0
Trinity(1)	10	2	1	1	2	1	1	50	46	82	0
Tulare(2)	253	13	1	2	4	1	3	43	39	75	0
Tuolumne(1)	22	2	1	1	2	1	1	48	44	80	0
Ventura (2)	267	14	1	2	14	1	2	55	51	0	0
Yolo (1)	78	4	1	1	6	1	1	50	46	82	0
Yuba (1)	31	2	1	1	2	1	1	57	53	0	0
Central DCSS	30	2	0		2	0	0	38	34	70	0
Central CCR	25	2	0		2	0	0	62	58	0	0
Model Office	20	2	1		2	1	1	39	35	71	0
Training Center Los Angeles	248	8	1		8	1	1	39	35	71	0
Training Center Sacramento	124	4	1		3	1	1	36	32	68	0
Training Center San Francisco	93	3	1		3	1	1	40	36	72	0
Training Center Redding in Shasta County	62	2	1		3	1	1	45	41	77	0
Implementation Total	**9,575**	**496**	**79**		**189**	**63**	**120**				
Implementation Total Refresh	**5,477**	**284**	**54**		**0**	**0**	**73**				

Figure 3C.13.4-8 Device Quantities for Rollout. *The alliance has gathered data from each site to plan the roll out for the CCSAS CSE application.*

Workstation quantities are derived from each site's staff count. Printer quantities are calculated on a ratio of 20 users per printer with a minimum of two printers per site for redundancy. The value of 20 users per printer is based upon the speed of the printer, ease of access, and load expectations, as measured by the anticipated print volumes. All local printers are 25 pages-per-minute duplex printers. The total annual local print volume is expected to be 140,000,000 pages. Router and switch quantities are based upon the networking infrastructure that is required to support the number of devices at each site.

The quantities of the image equipment including scanners, image servers and image workstations are derived from the estimated number of pages that will need to be scanned and processed per day. Only one image server is required per LCSA to keep up with the volume of images to be stored and forwarded. Redundant image servers are not considered necessary because current copies of imaged documents is not critical to the business process, if for some reason a repair had to be made in the event of a hardware failure to the image server. There is one image workstation for approximately every 5,000 pages per day.

3C.13.4.5 Risks and Mitigation Strategies

The alliance hardware and software installation approach takes into account an understanding of the key implementation risks specific to the CCSAS CSE environment and provides mitigation strategies for those risks.

Risk	Mitigation Strategies
Hardware and software rollout does not accommodate CCSAS complexities.	The alliance has custom-tailored the hardware and software installation strategy for the environment of CCSAS CSE while using proven and well-established practices. The IBM hardware and software methodologies, including the Network Deployment Methodology (NDM) and the Asset Management (AM) Methodology have been validated in hundreds of installations. These refined methodologies are the right match for large and complex rollouts. They are powerful contributors to a successful CCSAS CSE solution.
Site cannot accommodate new hardware installation.	The alliance will conduct site surveys early to identify and remedy any situations that would adversely impact the hardware and software installation timetable. Our cabling strategy is accomplished within a compressed six-month period beginning seven months prior to the first installation in Phase II. This allows a safety margin of time to deal with any issues that might delay cabling, including unanticipated office moves, the discovery of asbestos, or constraints on those facilities designated as historical landmarks.
Hardware is rolled out too soon for effective use.	The desktop installation strategy is done on a Just-In-Time (JIT) basis. Each location is scheduled for its desktop installations three months prior to the cutover date to the CCSAS CSE application.
Hardware is not rolled out soon enough.	Each location is scheduled for its desktop installs within three months prior to cutting over to the CCSAS CSE application. This allows ample time for user familiarization and the CCSAS CSE application training that precedes the cutover.
Project length of CCSAS results in out-dated equipment.	The CCSAS project extends longer than the life of the equipment that is initially deployed. This approach therefore includes the management and execution of tech-refreshes. A tech-refresh occurs when a piece of equipment is considered too old to continue to be of value and therefore needs to be retired and replaced. The tech-refresh cycle for desktops is three years. Tech-refresh intervals are established for peripherals, photo imaging, and printer equipment in accordance with CCSAS CSE application-approved criteria based on: • Depreciation schedules • End of the useful, supportable life of a piece of hardware, software, or license

Figure 3C.13.4-9 Hardware and Software Installation Risks and Mitigation Strategies. *The alliance has carefully considered the potential risks that could be associated with the CCSAS CSE HW/SW installation and has planned for their mitigation.*

3C.14.0 Production Support

For more than 40 years, alliance member IBM has had experience as a systems integrator and provider of production support for government customers. We understand the complexity of designing, developing and operating the CCSAS CSE system and have assembled a strong team to accomplish this endeavor. The depth and breadth of the alliance's collective experience will serve to lower the production support risk to the FTB, Department of Child Support Services and the Local Child Support Agencies in the development and operations of the California Child Support Automation System.

The alliance will be responsible for the operation and management of the CCSAS CSE infrastructure, server environment, and desktop operations to support CCSAS CSE project processing. The IBM e-business Hosting Center (eBHC) for CCSAS CSE is located at San Jose, CA. This responsibility will include establishing and maintaining a properly trained and staffed Hosting Center population, including the necessary management and support staff. The hours of operation of the Hosting Center will be 24 x 7 x 365 (except for scheduled downtime).

Figure 3C.14.0-1 The alliance's SMC Disciplines - *Our integrated set of processes and practices create a reliable and high- performance environment.*

The alliance's wealth of production support knowledge and experience is embodied in our SMC Disciplines, shown in figure 3C.14.0-1. These Disciplines will enable the alliance to plan, organize, measure, and control the delivery of CCSAS CSE services, lowering the risk of system outages and enhancing reliability. Through this series of

The alliance will use established processes to manage the production support activities for the CCSAS. Further, we'll provide the levels of availability and reliability that are necessary for a project of this importance.

processes and techniques, improved quality assurance and productivity will be achieved systematically.

We feel a special commitment to providing custom systems and solutions that perform consistently, are reliably available, and are quickly and easily enhanced to support changing policy and regulatory requirements. We have proposed a tightly integrated production support organization that will allow us to effectively support the CCSAS CSE environments and, by extension, the CCSAS CSE end users and customers. Below are the key components of our overall production support approach, which includes CCSAS CSE operations, maintenance and help desk. The alliance will use:

- **An Established Data Center** – The CSE system will use the IBM San Jose e-business Hosting Center, which will leverage the existing data center infrastructure and operational procedures. The San Jose site is a hardened facility with redundant power supplies, battery backup, and dual network connections.
- **Experienced Operations and Support Personnel** – The alliance brings a wealth of experience in the technologies, software, and tools we will be using for the CCSAS CSE environments. Alliance team member IBM has developed the hardware that will support CCSAS CSE. Our system programmers and production support personnel know the CCSAS CSE software environment. That advantage will enable us to not only fix many problems before they impact the end users, but also to quickly fix those problems that *do* affect the end users. The alliance's Service Delivery Center has a deep bench of technical resources that can be leveraged for the CCSAS CSE project.
- **Customer Service** - The alliance's Customer Service component will respond to public inquiries, from general information to specific child support questions. The alliance will conduct a Customer Service Support Center Assessment (CDL TM 072) that shall be used as the basis for the eventual Customer Service solution. The Customer Service Center will provide improved services to families with 24-hour self-service access.
- **System Management Controls** - The alliance will utilize its Systems Management Controls (SMC) to establish and maintain control of the CCSAS CSE environments. These SMC Disciplines form the basis for the Operations, Maintenance, and Help Desk components of CCSAS CSE production support. Systems Management Controls are integrated to deliver a consistent and measured approach to providing the services. Our SMC disciplines are the cornerstone of delivering system integrity and availability for CCSAS CSE. These processes and procedures have been defined and refined during thousands of projects by incorporating lessons learned. Systems Management also provides reports that are essential to managing and evaluating services and for determining production support areas that require focus and improvement. The alliance's SMC Disciplines include the following processes:

- **Service Level Management** - Service Level Management is the foundation on which delivery of Systems Management is built. Service Level Management includes regular reviews of Service Level Attainment Reports with the CCSAS CSE project to determine how well commitments are being achieved over time
- **Quality Control Management** - Quality Control Management means the alliance will plan, organize, execute according to plan, measure, correct, and monitor the technical infrastructure Service Delivery Environments
- **Operations Problem Management** – To identify, record, track, and correct issues that impact service delivery, recognize recurring problems, address procedural issues, and contain or reduce the impact of problems that occur
- **Operations Change Management** – Operations Change Management is a unique operational process. Operations Change Management is used to assess if the change is necessary, validate the adequacy of the acceptance test, schedule the promotion into the CCSAS CSE environments, notify the appropriate functional team, and verify implementation. Persistent operational problems can be escalated to the PMP Problem Management process
- **Recovery Management** - Recovery Management is the process of planning and establishing the recovery procedures that will be required to provide normal services to CCSAS CSE in the event of a failure
- **Offline Processing Management** – For controlling production offline processing work, including the scheduling of resources, the processing of data and transactions, and the distribution of data/information between users and facilities
- **On-line Processing Management** - On-line Management is the process of coordinating the appropriate personnel, information, tools, and procedures required to manage on-line networks (both remote and local), operating systems, and applications
- **Performance Management** – To monitor, measure, analyze, and report systems performance
- **Capacity Management** – For the development and maintenance of tactical and strategic plans to verify that the e-business Hosting Center environment accommodates growing or changing business requirements
- **Help Desk Management -** Detailed Help Desk procedures, plans and processes to support CCSAS end users and track problems to their resolution
- **Periodic Release Management Approach** – The alliance will manage, control, and implement system changes using a release-based approach. The design will build maintainability into CCSAS CSE right from the first design tasks, and use a release management approach to coordinate changes with the CCSAS CSE project staff during the planning, scheduling, and deployment of system changes.
- **Proactive Communication** – CCSAS CSE will provide users with up-to-date system status information about availability and the expected impact and length of any unplanned outages. The schedule for implementation of CCSAS CSE changes and enhancements, response-time expectations for processing different types of service, and the achieved service levels will be clearly stated and made available to the CCSAS project staff. We will use several tools and processes to communicate system status information and announcements to users. Those tools include system broadcast messages, e-mail communications, and direct phone calls to site personnel.

- **Centralized Service Desk** – The Help Desk will answer, respond to, resolve, and manage user requests. The Help Desk approach will provide users with a single point-of-contact for CCSAS CSE problems. Level 1Help Desk services are provided by either the county or the alliance, depending on the desktop option in place, as further defined in Section 3C.14.3. The Help Desk will perform the problem determination and, if a support request warrants it, will escalate the call, as necessary, to Level 2 and Level 3 support groups in order to resolve the problem. The Help Desk procedures include a closed-loop process to monitor call resolution.

Our production support strategy, as highlighted above, will lead to a high-quality service organization that can maintain system integrity and availability, and keep the CCSAS CSE application stable, responsive, and current in regard to policy changes. Our proposed operational methodology, support organization, and operational tools work together to provide optimal user satisfaction.

The remainder of this section describes the alliance approach to Customer Service.

3C.14.0.1 Customer Service

The alliance proposed the Customer Service solution described in this section. The alliance and State have agreed during contract negotiations that the requirements and scope for the Customer Service solution are not sufficiently well-defined to permit a full, pre-defined solution. Accordingly, the Customer Service Support Center Assessment (CDL TM 072) shall be used as the basis for the eventual Customer Service solution. That solution shall then be developed by the alliance using a level-of-effort approach that shall utilize the portion of the contract budget originally allocated to the Customer Service solution.

The State and alliance have agreed that the Customer Service solution shall follow a regionalized approach and shall accommodate call volumes for the CCSAS system (including IV-D and non IV-D payment, collection and disbursement inquiries). The alliance will provide management, oversight and technical support. The State will provide the regionalized call center staff, and participate and assist with the management, oversight and technical support.

The remainder of this section is left in place as representative of the originally proposed alliance solution.

The alliance, through IBM's extensive experience in Customer Service, has a proven method to develop a Customer Service solution that will:

- Provide effective and measurable efficiencies in service delivery over a range of delivery channels
- Provide effective and measurable improvements in customer satisfaction for case participants
- Interact effectively with other Child Support Program resources (e.g., Web sites, information sources)

- Provide a range of information and services for participants, lawmakers, interest groups, and other interested parties

The alliance's proposed Customer Service solution, shown in Figure 3C.14.0.1-1, provides enhanced self-service and agent-assisted Customer Service for the CCSAS case participants and promotes a more consistent, accurate delivery of business service assistance.

California's Child Support Program serves many different stakeholders – case members, employers, financial institutions, other states, caseworkers, and other third parties. Providing consistent Customer Service has proved to be difficult, given the current environment of multiple systems and localized delivery of services.

The alliance's Customer Service approach will provide multi-channeled access and a single point of contact for the Child Support Program stakeholders. The components of Customer Service are:

- Agent-Provided User Assistance (Call Center) – Child Support personnel will operate Help Desk and Call Center organizations to provide support to system users and external customers.
- Interactive Voice Response (IVR) – A telephone-accessed interface for case members to obtain information about their cases.
- eChild Support Customer Self-Service Application - Provides Web-based availability for child support stakeholders to perform self-service functions.

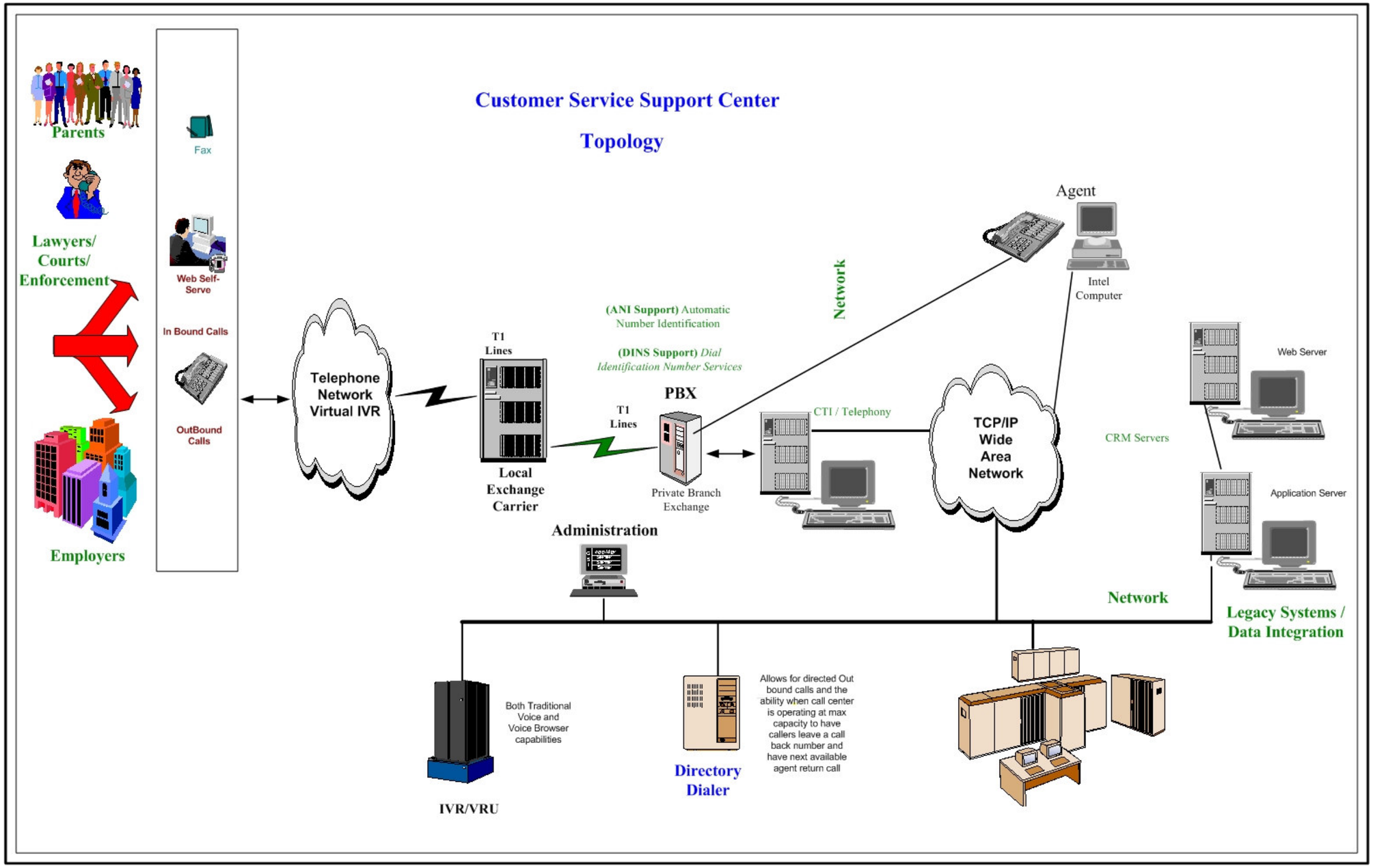

Customer Service Support Center
Topology
Parents
Lawyers/
Courts/
Enforcement
Employers
Fax
Web Self-Serve
In Bound Calls
OutBound Calls
Telephone Network Virtual IVR
T1 Lines
Local Exchange Carrier
(ANI Support) Automatic Number Identification
(DINS Support) Dial Identification Number Services
T1 Lines
PBX
Private Branch Exchange
Network
Agent
Intel Computer
CTI / Telephony
TCP/IP Wide Area Network
CRM Servers
Web Server
Application Server
Network
Legacy Systems / Data Integration
Administration
Both Traditional Voice and Voice Browser capabilities
IVR/VRU
Allows for directed Out bound calls and the ability when call center is operating at max capacity to have callers leave a call back number and have next available agent return call
Directory Dialer

Figure 3C.14.0-2 Customer Service Support Center – *The alliance's Customer Service Support Center will effectively answer CCSAS customer inquiries.*

Agent-Provided User Assistance (Call Center)

The alliance is proposing providing a Call Center with appropriate agent staffing and training to address a wide variety of customer issues. This Call Center will use Customer Relationship Management (CRM) software to provide: (1) scripting of call responses; (2) integrated automated user information access and updated functionality with the CSE system; and (3) help issue creation, tracking, and management.

The primary purpose of the Call Center will be to answer calls from child support case members and other stakeholders, as well as to provide information and perform appropriate case actions. The Call Center will also provide support on the use of the eChild Support Customer self-service application and IVR. The Call Center will be accessed via a single phone number for customers across the State. The Call Center will use a Tailored Response Management process using interactive scripting through the CRM tool to service common information or action requests. Call-tracking software will be used to monitor service and to provide statistical analysis of Call Center performance (e.g., the average calls per day, average time to complete a call).

When a call is received, the caller will interact with the IVR, which will gather appropriate information from the caller (e.g., the case number) and satisfy the user's question or request when possible. If the user elects to speak to an agent, the call will be transferred to an agent, along with the information that was entered by the caller. Call queues and routing will be tailored to promote the routing of a call to the first available agent or a Subject Matter Expert agent, based on the caller's needs. Automated account lookups and other CRM-to-CSE system integration will be done to support accurate and speedy handling of the call.

Interactive Voice Response (IVR)

The IVR component will provide informational and some tailored self-service capabilities for case members and other stakeholders. The alliance is proposing that IVR provide a subset of the services provided by the Web-based eChild Support application, specifically those whose workflow and complexity fit the usage characteristics of a speech or touch-tone-enabled user interface. In addition, the IVR will provide direct call routing to the Call Center to support more complex user needs. The alliance estimates that as much as 80 percent of user requests made by telephone will be able to be satisfied by the IVR, without agent intervention.

Functions provided by the IVR are:

- Dissemination of general information
- Request reviews and receive status information regarding case orders
- Options to set up electronic payments for a case

eChild Support Customer Self-Service Application

The alliance is proposing a strong set of self-service capabilities to support the child support case lifecycle and CCSAS project. We believe that comprehensive self-service capabilities will not only provide superior service for case participants, but will also serve to reduce the workload for Child Support Program workers as the case workload

expands. The alliance believes that the current economic climate will be a catalyst for a significant increase in new child support cases in the next three-year period. Strong, self-service capabilities will benefit the case participant, Child Support Program interested party, and, ultimately, the caseworker by reducing the need for manual case actions by caseworkers.

We recognize that many counties have existing child support Web sites that provide many valuable services. The eChild Support Customer self-service functionality will provide a gateway to those sites as a single point of access and will enhance the overall self-service functionality currently available. In addition, the eChild Support Customer Web site is the ideal vehicle for quickly disseminating new information to the child support community, providing broad reach while helping to mitigate spikes in the call volume to the Call Center.

Enabling stakeholders with any level of Internet expertise to navigate the self-service Web site will be key to its success. The eChild Support Customer Web site will provide multiple site navigation techniques, such as navigating the Website by organization, subject, or geographic area. For newcomers, the portal will provide customers with wizards to guide them through the services and functions available on-line.

Allowing access to different information for different types of users will be handled via *roles*. Roles will be used to group users with similar interests and target information for that group.

eChild Support Customer self-service puts information in the hands of parents when they need it – nights, weekends, and holidays. By accessing the Internet at home, work, or in libraries, clients quickly receive accurate information that was previously only available from staff during office hours.

Our Sizing Process

In order to appropriately size a Call Center solution that will meet the needs of the Child Support Program community, we have to begin with some basic assumptions about the end user population. The information we used and the assumptions we made for the purposes of this estimate were based on data from SCP Exhibit 3F. Additional sizing assumptions are located in the assumption section later in this section.

Beyond the sheer numbers of calls that a Call Center can be expected to receive over the course of a year, there are certain other variables that we must take into consideration as we plan. One of those variables involves some basic assumptions about the Call Center and the people who will work there. We assume that:

- Each Customer Support Representative will have 6 hours of productive work each day, five days a week
- During the average month, there are 22 working days for each Customer Support Representative
- There will be an average call time of 4.5 minutes, based on three minutes of talk time and 1.5 minutes of After-Call Work (ACW)

Using these assumptions, we can develop some baseline numbers for the volume of calls that a Customer Support Representative will handle during a six-hour day. The formulas we use to determine the baseline average numbers consider that there are:

- Six hours of productive work daily
- 6 hours % 60 minutes = 360 minutes per day
- 360 minutes % 22 days = 7,920 minutes per month
- 7,920 minutes + 4.5 minutes per call = 1,760 calls per month, per agent

We have built a worksheet to show the effect of the various assumptions on the required number of Call Center Agent staff. The scenario below uses the six-hours-per-day foundation for productive work for each agent, per day. Further, the scenario is based on current case types and caseloads with Non-IV-D cases, and with 30% of the volume being migrated to the Web application and another 20 percent decrease due to the demographic differences between Los Angeles and the rest of the state of California. The handling of all financial calls by the SDU is expected to reduce the volume by a further 80 %. (See the assumptions above.)

Customer Support Representatives

The following scenario worksheet shows the assumptions for the Customer Support Representatives.

Customer Service Assumptions	
Hours per day per agent	6.00
Minutes per Hour	60.00
Minutes per day for agents	360.00
Number of days per month	22.00
Minutes per month per agent	7,920.00
Length of calls in minutes	4.50
Number of expected calls handled by agent per month	1,760.00
Number of expected calls into the call center per month year 2005 Based on 25% of calls	29,865.00
Total number of estimated agents to handle volume year 2005	16.97
Number of expected calls into the call center per month year 2006 Based on 65% of calls	81,347.00
Total number of estimated agents to handle volume year 2006	46.22
Number of expected calls into the call center per month year 2007 Based on 100% of calls	130,837.00

Customer Service Assumptions	
Total number of estimated agents to handle volume year 2007	74.34
Number of expected calls into the call center per month year 2008	136,526.00
Total number of estimated agents to handle volume year 2008	77.57

Figure 3C.14.0-3 Customer Service Assumptions - *These Assumptions show the basis for the Customer Support Representatives.*

Planning

We will use the Statement(s) of Work (SOWs) and the Contract Deliverable List (CDLs) to guide our discussion of how the alliance plans to meet the various requirements that govern the design and development of the Customer Service Support Center.
Customer Service Support Center Assessment (CDL TM 072)

The alliance's Customer Service Support Center Assessment examines and reports on the current state of the Customer Service strategy, capabilities, enablers, and operations. The assessment includes a review of the access channels for service and/or support operations. The assessment is based on a structured process and on categorization of the components and activities associated with the Call Center. The assessment will identify the gaps between the current county Call Center standards and the best practices in the Call Center field. The Customer Service (Call) Center is examined across five key business areas:

- Organization
- Process
- Technology
- Information Access
- Facilities

In addition, we have developed assessment points with which we can evaluate the key business areas, looking for places where specific enterprise improvement can be made. Those assessment points are:

- Vision, Mission, Strategy, and Objectives
- Customer Management
- Channel Management
- Contact Management
- Staff Management
- Education and Training
- Data, Information, and Decisions
- Performance Management
- Communications Infrastructure
- Applications and Databases

The key benefits of the Customer Service Support Center Assessment model are:

- Gaps in the alignment of Customer Service goals and objectives, initiatives, and strategy can be identified
- Current operations across all the facets of Customer Service can be clearly defined
- An understanding of the defects and opportunities for improvement can be gained
- Projects and recommendations for Customer Service strategy, goals, and objectives can be developed

This Customer Service Support Center Assessment effort may necessitate changes to the business requirement baseline; all changes will be conducted in accordance with an approved, documented Change Control process.

Customer Service Support Center Plan (CDL TM 073)

The next phase of the process is to develop the Customer Service Support Center Plan. We will take the information and recommendations we gather from the Customer Service Support Center Assessment and apply them to developing the definitions for the Customer Service Support Center requirements as the foundation for the Plan, which will, in turn, provide the roadmap for developing the Customer Service Support Center. The Plan will cover each area of Customer Service: the Call Center, IVR, and Web-based self-service. The Plan will define the requirements for each of these components and will provide information describing how those requirements will be met.

Following a joint meeting(s) to define Customer Service Support Center strategies, policies, and procedures, a document will be drawn up to capture the results of Assessment review and to finalize any Change Orders to the bid, based on the Assessment's recommendations in any of the five areas listed below:

- Organization
- Process
- Technology
- Information Access
- Facilities

The Customer Service Support Center Plan will contain the plans for:

- Facility
- Staffing
- Call Center Procedures and processes
- Telecommunications hardware and software
- Voice Response Unit hardware and software
- Call Center Applications hardware and software
- Web-based self-service
- Transitioning

The steps involved in this defining the plans will contain the following processes as needed;

- As-Is Process – This will consist of the business functionality as described in this document
- To-Be Process – This will take the business processes that were agreed upon after the Customer Service Support Center Assessment was reviewed and the merge them with the AS-IS-Process document to produce the To-Be-Process
- High Level Gap Analysis – This step will take the output of the prior step and match to the operational level processes from the selected CRM package to identify any gaps
- Requirements and Analysis – This will involve taking the gaps in operational level processes from the previous step to build a set of requirements for Customer Service Support Center
- The result of the prior steps is the roadmap for Implementation Phase of the Customer Service Support Center

The Customer Service Support Center Implementation Phase

The alliance's project implementation approach for the Customer Service Support Center incorporates Rapid Application Design techniques with package implementation capabilities and IBM's proven Project Management Method model

Supporting the methodology is a complete, life cycle Project Management effort and Quality Assurance Review process that allows us to monitor the progress toward a successful delivery of the Customer Service Support Center.

Based on the alliance's understanding of current (CSE) environment and based on our experience automating Customer Service solutions, we will use our implementation approach to configure the Customer Service Support Center Solutions.

The alliance's methodology will also reduce the overall complexity and risk of the project by:

- Dividing the critical path development into manageable component deliverables
- Developing internal skills for the solution set through knowledge transfer
- Supporting continuous feedback and review

The methodology is comprised of three phases: Launch, Configure and Rollout. Throughout the phases, Project Management and Quality Assurance Management are continually involved to produce the best possible solution for the CCSAS CSE project.

The alliance's methodology is based around work products, developed in five domains that comprise critical areas that a project will address:

- <u>Business</u> – addresses business transformation that leverages the best practices supported by the Siebel software, and yet is constrained by the capabilities of this software package

- Organization – addresses change management, organizational design and implementation, and education and training
- Application – addresses the design, configuration, testing, and deployment of the software solution, development of interfaces to legacy and other systems, including integration and system testing of the complete solution
- Architecture – addresses the assessment of existing infrastructure, the design of the new infrastructure, and the processes and procedures to run it
- Project and Quality Assurance Management – addresses deployment and other aspects of project including Quality Assurance Management specific to system integration and consulting

Launch

The Launch Phase utilizes a rigorous management process to complete a requirements definition. The core of this exercise is a Fit/Gap Analysis in which business processes are defined, needs are mapped directly to standard product functionality as delivered with the vendor software, and where organizational, interface and technology requirements are documented.

Configure

During the Configure Phase, the alliance will focus on:

- Configuring and testing the software
- Resolving gaps between the Requirements and asset capabilities
- Developing programs
- Infrastructure design and implementation
- Organizational redesign
- Management of business transformation and change issues

During application configuration period, configuration of the solution components data conversion, interface processes, report programs, and security profiles will be developed and tested concurrently, followed by final system testing and end-user training.

Final system testing will consist of the following activities:

- Development of conversion procedures and programs
- Creation of interface programs
- Generation of sample reports
- Execution of volume and stress testing
- Completion of final user acceptance testing

Development of a go-live strategy that will specifically identify the data-conversion strategy, initial audit procedures, and a rollout team support structure. The final step in the Configure Phase will be to approve and pilot the solution, and check the readiness of the Customer Service Call Center solutions to go live.

Rollout

During the Rollout Phase, the alliance will deploy the configured Customer Service Support Center solution that has been accepted by the CCSAS project staff and the user community. Deploying the solution may entail a variety of activities:

- Installing any as-yet uninstalled infrastructure components

- Migrating data over to the new system
- Training and supporting the extended community of users
- Implementing the new organization design and performance management systems
- Carefully managing the change

Once the deployment preparation work has been completed, the Customer Service Support (Call) Center's readiness to go live will be confirmed and the system will be officially turned on.

Immediately following rollout, the solution software will be reviewed and refined to validate that the business environment is fully supported. This will involve verifying the accuracy of the business transactions, seeing to it that the data and the system have been successfully cut over to the production environment, and verifying that the CCSAS end-user community's needs have been successfully met.

Training Customer Service Support Center Training Materials CDL TM 074

The alliance will develop training materials and programs that teach and reinforce Call Center specific skills, and can be customized to address specific performance requirements, issues, or deficiencies. The goal is to create training that provides the trainee with an accurate simulation of the actual job responsibilities.

The alliance's instructional design process is built on the belief that quality performance is the goal of effective training. We will use the five-step ADDIE (Analysis, Design, Development, Implementation, and Evaluation) instructional design model to develop the training programs. This proven methodology is applicable across a variety of training scenarios and audiences. Training objectives, content, evaluation strategy, and delivery are all factored into decisions regarding the most effective media and methods.

The training delivery can include paper-based documents and training manuals, reference materials, job aids, overhead/slide presentations, video presentations, role-play scenarios, flip charts, whiteboard, hands-on demonstrations, and discussion groups.

The alliance will design and develop interactive technology-based learning solutions to meet CCSAS project staff's specific learning objectives and performance goals. Technology-based training from the alliance will include interactive Web-based courses, online learning modules, customized computer knowledgebase's, and e-mail flashes.

The alliance will also develop ongoing training programs to fulfill Call Support Center needs. The amount and type of training will vary according to several factors, including business needs, the skill levels of the customer service representatives (CSRs), and the call volumes. The alliance CSRs typically receive about four hours of ongoing training per month, which includes:

- Two hours of pre-shift training
- One hour of supervisor meeting and development time
- Daily pre-shift meetings and weekly tailgate training sessions to discuss changes impacting the project and to deliver appropriate instructions regarding changes

Staffing

The staffing of the Customer Service Call Center will be structured such that the CCSAS project staff will provide 75% of the staff for the Call Center and the alliance will provide 25%. We have used the following assumptions to develop our staffing requirements:

- Fifty percent of all calls will be handled by the IVR
- Initially, 50 percent of all calls will be resolved on a first call basis; this number will be increased as experience with the systems and processes increase
- Staffing will be in line with peak demand periods

The following table shows the staffing positions by year projected for the call center:

Customer Service Contact Center Staffing							
Position Description	**Skill Sets**	**Head Count year 1**	**New head count for year2**	**Total Head Count year 2**	**New Head count year 3**	**Total Head Count year 3**	**Final Head Count**
CCSAS Management	**CCSAS project Resource**						
Call Center Manager	Knowledge of call center processes, metrics and technology, Personal Management skills, Child Support knowledge, Executive skills 10+ years of Call Center management exp.	1	0	1	0	1	1
Call Center Agent Manager	Knowledge of call center processes, metrics and technology) Personal Management skills, Child Support knowledge 5+ years of Call Center management exp.	1	0	1	0	1	1
Call Center Agent Training Manager	Knowledge of call center processes, metrics and technology) Personal Management skills, Child Support knowledge, Training development experience, 5+ years. Training delivery experience	1	0	1	0	1	1
Call Center Quality Assurance Manager	Knowledge of call center processes, metrics and technology) Personal Management skills, Child Support knowledge, 5+ years of Call Center QA experience	1	0	1	0	1	1
Call Center Operations Support Manager	Knowledge of call center processes, metrics and technology) Personal Management skills 5 to 7 years experience Supporting Networking, Hardware, Software and Telephony systems	1	0	1	0	1	1
Call Center Facility Manager	Call Center Facility management experience 5 + years of facility management experience	1	0	1	0	1	1
Subtotal Managers		**6**		**6**		**6**	**6**
CCSAS Call Center Staff	**CCSAS project Resource**						

Customer Service Contact Center Staffing							
Position Description	**Skill Sets**	**Head Count year 1**	**New head count for year2**	**Total Head Count year 2**	**New Head count year 3**	**Total Head Count year 3**	**Final Head Count**
Call Center Agent Team Leads	Knowledge of call center processes, metrics) Personal Management skills, Team leadership skills 5 to 7 years in child support Call Centers	1	2	3	2	5	7
Senior Call Center Agents	Knowledge of call center processes, metrics) Personal Management skills, Team leadership skills 3 to 5 years in child support Call Centers	2	3	5	5	10	12
Call Center Agents	Knowledge of call center processes, metrics) Personal Management skills, Team leadership skills 2 to 3 years in child support Call Centers	2	8	10	19	29	46
Call Center Agents (Junior)	Entry level	8	9	17	2	19	19
Subtotal Staff		**13**		**35**		**63**	**84**
CCSAS Operations Support Staff	**CCSAS project Resource**						
Quality Assurance (Senior)	Knowledge of call center processes, metrics) Personal Management skills, Team leadership skills 5 to 7 years in child support Call Centers	2	0	2	0	2	2
Quality Assurance (junior)	Knowledge of call center processes, metrics) Personal Management skills, Team leadership skills 3 to 5 years in child support Call Centers	0	1	1	1	2	2
Call Center Trainer	Knowledge of call center applications, Knowledge of training development methodologies. Teaming skills 3+ years in Call Center training	1	0	1	2	3	4
Call Center Technical Support	Knowledge of call center hardware, networking and telephony skills, 5 to 7 years experience Supporting Networking, Hardware, and Telephony systems	0	0	0	0	0	5
Call Center Internal Help desk staff	Knowledge of call center software , 5 to 7 years experience Supporting Call Center applications(Siebel), CTI Software, and Telephone software systems (PBX, Nice, Blue Pumpkin, etc)	0		0		0	3
Administrative Assistants	PC and office skills	0	1	1	1	2	2
Subtotal OSC Staff		**3**		**5**		**9**	**18**

Customer Service Contact Center Staffing							
Position Description	**Skill Sets**	**Head Count year 1**	**New head count for year2**	**Total Head Count year 2**	**New Head count year 3**	**Total Head Count year 3**	**Final Head Count**
CCSAS Total Staff	**CCSAS project Resources**	**22**		**46**		**78**	**108**
Alliance Management	**Alliance Resources**						
Call Center Project Executive Manager	Call Center management (i.e. strong understanding of call center processes, metrics and technology) Personal Management skills, Child Support knowledge	1	0	1	0	1	0
Call Center Operations Support Manager	Knowledge of call center processes, metrics and technology) Personal Management skills 5 to 7 years experience Supporting Networking, Hardware, Software and Telephony systems	1	0	1	0	1	0
Subtotal Managers		**2**		**2**		**2**	**0**
Alliance Call Center Staff	**Alliance Resources**						
Call Center Agents	Knowledge of call center processes, metrics) Personal Management skills, Team leadership skills2 to 3 years in child support Call Centers	4	7	11	8	19	0
Subtotal Staff		**4**		**11**		**19**	**0**
Alliance Operations Support Staff							
Call Center Trainer	Knowledge of call center applications, Knowledge of training development methodologies. Teaming skills 3+ years in Call Center training	1	0	1	0	1	0
Call Center Technical Support	Knowledge of call center hardware, , 5 to 7 years experience Supporting Networking, Hardware, and Telephony systems	2	1	3	2	5	0
Call Center Internal Help desk staff	Knowledge of call center software , 5 to 7 years experience Supporting Call Center applications(Siebel), CTI Software, and Telephone software systems (PBX, Nice, Blue Pumpkin,etc)	2	2	3	0	3	0
Subtotal OSC Staff		**5**		**7**		**9**	**0**
Alliance Total Staff	**Alliance Resources**	**11**		**20**		**30**	**0**
Total Staff		**33**		**66**		**108**	**108**

Figure 3C.14.0-4 Customer Service Contract Center Staffing – *Yearly projected staffing positions for the call center.*

Customer Support Business Functions

The following table summarizes the common functions that will be built as the core delivery capabilities for Customer Service. The table shows what capability will be available and by what delivery mechanism – Agent-assisted, IVR, and Web-based self-service. Detailed description of each function follows the table.

Customer Support Activity Description	Customer Support Representative	IVR	Web
Update member addresses and employee Information	Yes	No	Yes
Provide SSN, DOB, demographic information for Locate activities	Yes	No	Yes
Request a review of child support orders	Yes	Yes	Yes
Review case activity (significant events, i.e., NCP successfully located, paternity established, supported established)	Yes	No	Yes
Set up automatic deposit of support payments	Yes	No	Yes
Enroll in automatic payments	Yes	No	Yes
Process subscription service requests (i.e., subscribing to electronic billing or notification)	Yes	Yes	Yes
Update third party information	Yes	No	Yes
Apply for Child Support services	No	No	Yes
Read laws and requirements	No	No	Yes
Request call back from Case Worker	Yes	Yes	Yes

Figure 3C.14.0-5 Core Delivery Capabilities for Customer Service

Update Member Addresses and Employee

Authorized and authenticated case members will be able to provide updated address information for case members as well as information about current employer and employer contact information.

Provide SSN, DOB, Demographic Information for Locate Activities

Authorized and authenticated case members will be able to provide updated demographic information (like social security number, date of birth, etc.) about absent, non-custodial parents to assist in the locate process.

Request a Review of Child Support Orders

Authorized and authenticated case members will be able to view and print child support court orders and other appropriate court documents, if they have been entered into the CCSAS system.

Review Case Activity

Authorized and authenticated case members will be able to view historical case events, such as case application date, case creation date, or locate request submission date. The number and selection of events to show will be based on milestones that are significant to the case member but that will not reflect too intricate a level of detail on internal case processing activities or compromise confidentiality.

Set Up Automatic Deposit of Support Payments

Authorized and authenticated custodial parents will be able enter bank information (e.g., bank name, account number, routing information) to setup automatic account deposits to those accounts for payments of child support obligations.

Enroll In Automatic Payments

Authorized and authenticated non-custodial parents will be able enter bank information (e.g., bank name, account number, routing information) to setup automatic account withdrawals to pay child support obligations.

Process Subscription Service Requests

The ability for a customer to get bills, electronic documents and billing statements through email or over the web instead of paper and mailed, for example, instead of getting a monthly bill, the customer could get a monthly email reminder. These services would be able to be requested individually (in other words, you could sign up for just one, or all of the subscription services).

Update Third Party Information

Authorized and authenticated case members will be able to provide demographic, role, and contact information on third parties involved in their case, such as attorneys or employers.

Apply For Child Support Services

Custodial parents may complete a child support application on-line.

Read Laws and Requirements

Case members and other interested parties may browse or print child support laws, requirements, and policies from the Web site.

Request Call Back from Case Worker

Case members can log a request for their assigned caseworker to call them.

Metrics Reporting

Customer Service metrics-gathering mechanisms will be built into the Customer Service channel mechanisms – IVR, the Call Center, and the Web site – to gauge the effectiveness, quality, and coverage of the Customer Service Support. The object of gathering and reporting on such metrics will be to measure the effectiveness and user acceptance of each service delivery channel. Further, such information will lay the

foundation for planning future enhancements of specific channels and present the opportunity remedy any identified shortfalls.

While specific metrics will be determined during the design phase of CCSAS CSE, potential future metrics could include:

- Total inquiries/service requests by channel, which would gauge how effective each channel is and identify trends in each channel's usage
- Types of inquiries/service requests by channel, which will promote refining the type of functionality for each channel and open the door for taking advantage of future technology delivery capabilities
- Inquiries/service requests, which will determine what the most requested actions or questions are. This will help to both discover additional needed functionality and target how to plan future application modifications for noticing
- Number of repeat inquiries by case or group of cases, which may show if there are training or management issues that should be addressed regarding certain kinds of child support cases

Call Center Management Reporting

Call Center Management Reporting will be used to gauge the efficiency and effectiveness of the Call Center – how it handles Help Desk issues and external customer calls. These metrics will help identify areas where improvements can be made in issue processing, client interaction scripts, and Call Agent training. Like the metrics gathering and reporting discussed in the previous subsection, the actual Call Center metrics list will be determined during the course of Customer Service Center rollout, but potential metrics could be:

- Calls Answered
- Number and Percentage of Calls Transferred
- Number of Service Requests Opened by Type
- Number of Service Requests Opened by User
- Number of Service Requests Opened by Contact
- Number of Service Requests Opened by Channel
- Breakdown of Service Request Type by Customer
- Breakdown of Service Request Type by Contact
- Average Time for Service Request Resolution
- Average Time for Case Resolution by Customer
- Service Request Status Analysis (Number Open "dispatch", Open "assigned", Open "in process", Open "resolved", Closed, Cancelled)
- Status of Service Requests by User
- Percentage of Total Service Requests Opened by Group
- Percentage of Total Service Requests Resolved by Group, Team and Individual
- Percentage of Total Service Requests Opened by Area
- Number of Service Request Status Queries via Self-service Tools
- Number Of Updates To User Opened Service Requests via Self-Service Tools
- Number Of Service Requests Avoided By self-help Mechanisms and Resources
- Number of Service Requests Opened and Length of Time Opened By Priority

- Number of Service Requests Opened and Length of Time Opened By Type
- Number of Service Requests Opened and Length of Time Opened By Severity
- Number of Service Requests by Type (Day / Week / Month / Quarter / Year)
- Number of Service Requests by Severity (Day / Week / Month / Quarter / Year)

Hardware/Software System Environment

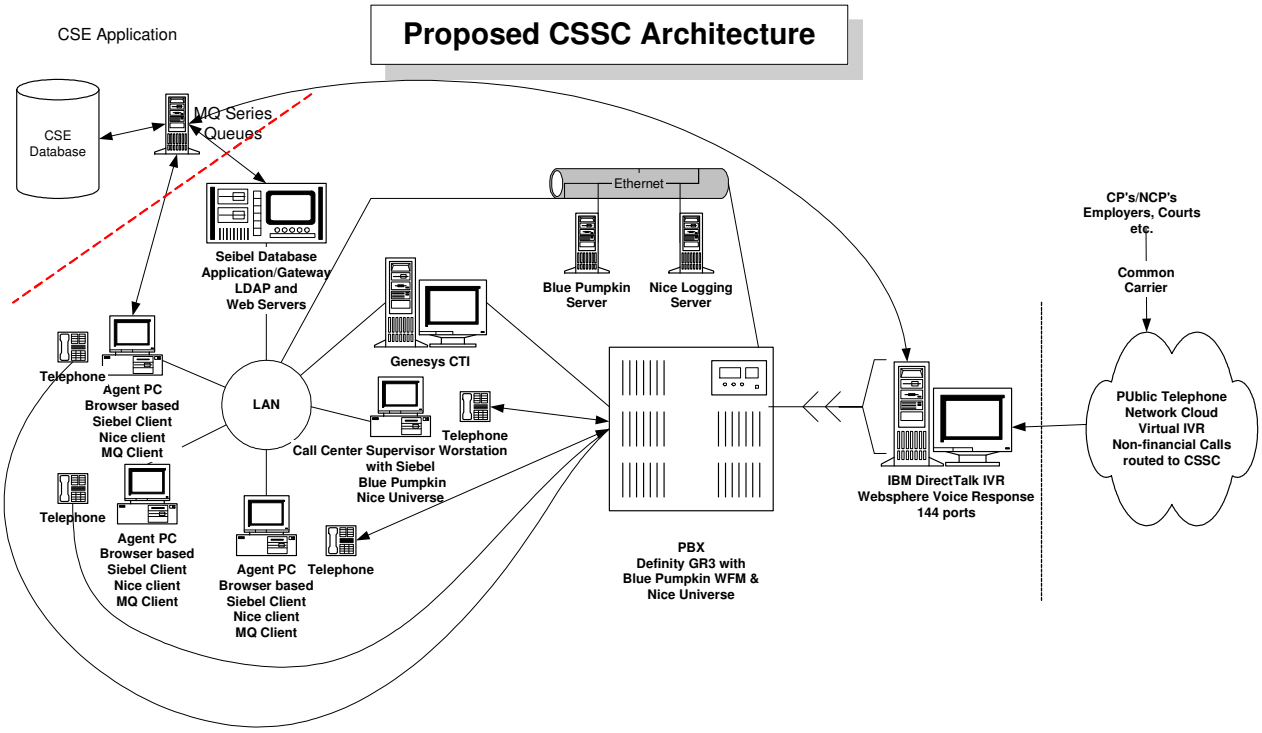


Figure 3C.14.0-6 Proposed Customer Service Support Center Architecture - *This architecture diagram shows the key components used to build the Customer Service Support Center.*

The following hardware/software list is provided for informational purposes:

Hardware/Software Equipment and Specifications	
CRM	Siebel Call Center V 7.00 Siebel SmartScript
CTI	Genesys Enterprise Routing 6.1
PBX	AVAYA DEFINITY G3 ECS
IVR	DirectTalk Software (DirectTalk renamed WebSphere Voice Response)

Figure 3C.14.0-7 Hardware/Software Equipment and Specifications

CRM

Siebel Call Center v7

The Siebel Call Center 7 solution provides agents with a unified, multichannel desktop view that consolidates all customer interactions, allowing them to easily access the critical information needed to manage customer requests. Siebel Call Center 7 also provides real-time analytic and reporting capabilities to support informed strategic and operational decisions. In addition, Siebel Call Center 7 provides full support for best practices and offers tools to help agents improve customer satisfaction.

Siebel SmartScript

Siebel SmartScript is a robust, workflow-based, dynamically generated user interface that assists the agent in asking the right question to the right customer at the right time. Based on the active question within the script, SmartScript guides the agent through a decision tree and selectively displays questions to be asked.

Siebel SmartScript Designer allows the management of scripts using an intuitive graphical user interface by providing a visual tool for creating and viewing the call script flow.

CTI

The selected CTI solution (Genesys Enterprise Routing 6.1) provides the contact center with the ability to provide fast and flexible customer service using a single universal queue to handle calls. This is accomplished within an integrated environment that allows users to create, modify, load, and execute routing strategies for efficient and effective interaction routing. Using the reporting applications CC Pulse and CC Analyzer, you can also generate reports on interaction statistics.

Enterprise Routing tracks an agent's skills and real-time availability. It provides agent-level, skill-based routing of customer interactions. Based on routing rules designed for specific business needs and objectives, the solution intelligently routes interactions to the most appropriate agent in the shortest possible time.

Telephony

The selected PBX chosen for use in the Customer Support Service Center is the Definity G3R Call Processor with Call Center Elite, Call management system and Intuity Voice Mail.

We have also selected the following software to work with the switch:

- Nice Universe –This is QA voice recording package that allows the monitoring of calls for quality assurance and training purposes.
- Blue Pumpkin Workforce Management – This package is used by the call center management to manage the utilization of call center personnel based on call volume history.

- 2 Wallboards – These display boards are used to provide statistics and messages to the call center personnel about call volume wait times etc. These wallboards utilize Ultralink CMS interface software.

IVR

The select IVR, IBM DirectTalk, is a versatile, powerful voice processing system. It allows callers to have direct access to the services and information they need, 24 hours a day, 7 days a week. DirectTalk allows contact centers to answer and screen a large number of IVR calls simultaneously and promptly, reducing caller wait time and improving customer satisfaction.

DirectTalk provides two-way speech capability once the desired information has been located, DirectTalk can audibly speak it to the caller, and using DirectTalk's advanced text-to-speech capability. This eliminates the need for pre-recorded, "canned" messages, and offers the flexibility of providing an endless variety of dynamic, real-time information on demand.

3C.14.0.2 Conclusion

The alliance will implement a complete, integrated Customer Service solution.. This call center, based on state of the art call center technology and applications, was designed to meet the current needs of the California child support system and have the growth capability to meet currently anticipated future needs. The alliance Customer Support Service Center will be the Custodial Parents and Non-Custodial Parents point-of-contact for non-financial Child support Case information.

CCSAS CSE operations will be delivered from the IBM San Jose e-business Hosting Center (eBHS). The alliance will use proven technical methods for the management and operations of the CCSAS through the use of System Management Control (SMC) Disciplines, our experience with e-business application hosting technologies, and traditional data center operations experience.

3C.14.1 Operations

The alliance knows that e-business hosting operations involve much more than managing business applications to provide availability and responsiveness. Increasingly, e-business hosting also involves handling dynamic data, electronic commerce, and customized services. It must be integrated with the enterprise's entire e-business strategy. For CCSAS CSE, hosting operations means being able to rely on 24 hour self service access to improve service to families and improve caseworker effectiveness. The alliance's production support capabilities will enable the state to concentrate on managing the CCSAS CSE program, and avoid having to worry about the underlying technology.

The alliance's breadth and depth of resources enables us to provide total Web-based e-business solutions from a single source, as just another advantage of this strong team. It enables us to implement, support and run powerful e-business solutions, Web hosting, electronic commerce, messaging, and collaborative solutions.

The CSE system application servers are located in "hardened" data center facilities that are linked to the global backbone networks and to the Internet with high-speed connections for performance. Our e-business Hosting Centers, designed with redundant facilities, are monitored 24 hours a day, 7 days a week, to provide continuous performance. The Hosting Centers' robust architecture is built for high availability with minimal or no single point of failure in our eBHC infrastructure. We understand the importance of the CSE system. With that in mind, it's good to know that our proven operations processes and the San Jose e-business Hosting Center provide significant protections against natural and man-made disasters.

The alliance has experience delivering high-volume applications like the CCSAS system.

The CCSAS project staff will have 24 x 7 escorted access to the San Jose eBHC and escorted access to the IBM Service Delivery Center in Boulder Colorado, allowing California State employees access to our facilities and staff for project management and oversight.

The alliance will deliver operations from the San Jose e-business Host Center, which is a hardened facility proven to provide system reliability and integrity.

While many service providers have been unable to meet customers' increasing scalability demands, IBM—an alliance team member— has accommodated clients' requirements for increased capacity with both ease and speed. In the last year, IBM hosted a variety of high-volume Web sites, including the U.S. Open, Wimbledon, the Grammy Awards, and the 2000 Sydney Olympics Web sites. The Wimbledon Web site was a record breaker with 2.3 billion hits over the course of the 14-day event and more than 960,000 hits per minute at its peak. Not only was the Wimbledon Web site the most visited official Grand Slam site in history, it set a new record for IBM-powered sports sites. In September 2000, this record was easily surpassed by IBM's site for the Sydney Olympics, which handled over 11.3 billion hits over the course of the Games, peaking at 1.2 million hits a minute. More importantly, the entire Olympics Web hosting environment was 100% available for the entire 17-day event.

Success stories like these illustrate the robustness and flexibility of alliance team member IBM's solution architecture and our ability to respond to both daily changes in Web site content *and* to dramatic surges in usage.

Alliance team member IBM provides Web server management and content hosting for over 10,000 users. Our services support a wide range of commercial e-business applications, such as secure online transactions (retail sales, stock trades, electronic banking, and airline reservations), inventory control, purchasing management, integration of Internet information with back-end legacy systems, inventory information, and interactive media.

The alliance will commence full operations responsibilities concurrent with the first release into production, and earlier for those system environments that are required to support the CSE application design, development and testing environments. The alliance will maintain and operate the CSE system for two years following full system implementation.

3C.14.1.1 Methods, Processes and Procedures

The alliance's managed e-business Hosting Services range from e-business *development and design* to *maintenance and security*. The goal of the alliance's operational methods, processes and procedures is to establish and maintain system integrity and availability.

The alliance provides a full service Production Support offering through our e-business Hosting Center in San Jose, California.

We handle server maintenance and support for a wide range of platforms and server configurations, making our services easily scalable and upgradeable. The alliance-managed servers offer secure, high-speed Internet and intranet access to vital business applications. The alliance provides an environment dedicated exclusively to the CCSAS Project that will enable the CCSAS project staff to run those vital business applications and conduct online transactions confidently, according to the Service Level Objectives. The alliance will maintain the CSE production environment to deliver system availability that meets or exceeds 99.5%.

These services are integrated within a corporate-level alliance framework, providing a suite of available hosting services. The highlights of those services are:

- **Hosting Center Management.** The alliance's Management Team uses rigorous process disciplines to oversee our Hosting Center operations and infrastructure. These disciplines are the pillars of our support methodology.
- **Server Monitoring.** The alliance uses a number of tools to monitor the hosting environment. Tivoli and a wide array of Performance and Problem Management tools enable the alliance to manage and monitor the health and status of the server environment in real-time.
- **Hosting Center Connectivity to the Internet.** The alliance connects users to the Internet via major and local Internet network providers that are capable of supplying multiple, redundant, high-bandwidth (DS3 or OC3) connections to the Internet from our Hosting Center locations around the world.
- **Disaster Recovery.** The alliance will customize a disaster recovery solution that extends the level of redundancy necessary to meet the CCSAS project's requirements. Our customized solution will include offsite tape storage and the ability to use alternative alliance sites as "recovery sites".
- **Backup and Recovery Management.** IBM's Tivoli Storage Manager (TSM) offers the capability to backup information from more than 30 multi-vendor platforms and supports a wide array of storage devices. The alliance has designed storage area networks (SANs) and Redundant Array of Independent Disks (RAID) capability into our hosting solutions to protect users from data loss in the event of a failure. The alliance typically performs an initial full backup, followed by daily incremental backups.

The key to the successful execution of the production support methods, processes and procedures is leadership and communication. The alliance Service Delivery Manager and the Operations Lead are responsible for production support performance including problem and service level management. Communication between the alliance members, the CCSAS project staff and production support team is an important component of identifying and resolving those issues that affect CSE operations and production support.

Service Delivery Manager

The Service Delivery Manager will be responsible for the overall quality of service delivery and long term strategic planning associated with that delivery. Additionally, the Service Delivery Manager will work with the appropriate CCSAS project staff to understand your strategic business requirements and enable the appropriate service resources to be available and deployed effectively to achieve CCSAS objectives. The Service Delivery Manager will report into the CSE Project Executive. The Service Delivery Manager's responsibilities include:

- Maintain positive customer satisfaction with the alliance's service delivery
- Provide the alliance Project Executive with single point of contact into service delivery
- Be onsite in the alliance Project Management office to keep abreast of project activities and be able to act quickly to resolve production support problems
- Own delivering against defined performance standards
- Own overall service quality
- Participate in account planning and strategy
- Identify and allocate technical professionals from the IBM Service Delivery Center to resolve critical situations or production support problems with a high severity level
- Represent the CCSAS project staff to the alliance production support delivery team
- Be a key resource to oversee a smooth transition of operational responsibilities to the State at the end of the contract

Operations Lead

The Operations Lead assists the Service Deliver Manager in the technical management of the CSE operational environment and has the following roles and responsibilities:

- Reports directly to the Service Delivery Manager
- Be onsite in the alliance Project Management office to keep abreast of project activities and be able to act quickly to resolve production support problems
- Support delivering against defined performance standards
- Monitor the technical aspects of our service delivery including operations problem management and service delivery management
- Assist with the resolution of severity one operational problems
- Provide technical design support as required

The alliance's eBHC Architecture

The alliance's eBHC (e-business Hosting Center) architecture is at the heart of our e-business hosting services. The scalability and flexibility inherent in the e-business Hosting Center design and operation reflect alliance team member IBM's long heritage hosting our users' vital mainframe applications. Each e-business Hosting Center has been designed to provide a secure and reliable hosting environment for servers and the

applications they support. They have also been designed to be fault tolerant, with no single point of failure, and to be capable of recovery in the event of an outage.

Designed for a Changing Business Environment

IBM designed the eBHC architecture as an alternative to the traditional Token Ring or Ethernet-based architectures that are typically used to host Web content. The eBHC architecture was designed to accommodate a heightened level of scalability, flexibility, and availability — essential in a global Web environment. These design objectives produced a Hosting Center infrastructure that:

- Allows the flexibility needed to adapt to different customer requirements
- Enforces the segmentation of each customer's Web environment and reduces the possibility of attacks occurring *within* the Hosting Center
- Includes a backbone network, enabling single hop inter-switch communication
- Provides complete redundancy, eliminating total customer site failures due to infrastructure outages
- Provides secure access to customer equipment for support

The Hosting Center's physical infrastructure consists of a complex array of cabling and shelving and a logical infrastructure that uses redundant Cisco or Xylan switches as the vehicle for connecting to the Internet, as well as to our customers' intranets and extranets. A redundant and fully meshed Asynchronous Transfer Mode (ATM) backbone enables a single hop between switches in the Hosting Center. Each edge switch will have a DS3 or OC3 connection to each of the ATM core switches, which are capable of supporting the traffic in the Hosting Center without degradation should a switch failure occur elsewhere.

Physical Design

The physical design of the e-business Hosting Center contains specially designed racks and shelving, and pre-configured cabling and wiring for a specific number of servers. The cabling to the racks is run in pairs to provide redundancy. The Hosting Center is pre-wired in order to facilitate the addition of servers as our customers' Web site traffic increases and their server requirements escalate. With cabling and power requirements met during the initial build-out, the Hosting Center is scalable and capable of accommodating urgent requests for expansion.

Adding servers to the Hosting Center is simply a matter of locating the next available shelf location, placing the server, and attaching it to pre-existing cables and power sources. Additional network connectivity and operating systems can also be quickly enabled via software loads and updates to the configuration database. The simplicity of these activities is made possible by the logical design of the eBHC architecture.

Logical Design

The alliance has connected the switching and server infrastructure in the form of a virtual Local Area Network (VLAN). The VLAN environment isolates and protects one customer from another, while enabling authorized traffic to flow to its proper destination.

A VLAN equates to a subnet, otherwise known as a *broadcast domain*. Each customer is placed within a distinct VLAN. The customer's high bandwidth traffic, such as broadcast or multicast traffic is contained within that VLAN and does not enter into the VLANs of other users. VLAN segmenting enables IBM to create a virtual private LAN for each customer's traffic. The alliance's password access controls and our automated audit of server traffic will further preserve the security and integrity of the CSE environments.

Firewalls are configured to filter the traffic between the VLANs, administrative traffic, incoming traffic from the Internet, private intranets, and extranets. VLAN network interconnections are continuously scaled to allow for throughput at speeds greater than 300 Mbps, and the switch backplanes are designed to accommodate extremely high (gigabit) bandwidth requirements.

The eBHC architecture enables customer traffic to be separated onto the VLANs after it is delivered to the Hosting Center from a major Internet backbone network provider or local Internet Service Provider (ISP). Each VLAN is dedicated to a particular customer's server. For example, the back end of the Web server is connected to the customer's data server via a dedicated VLAN, and the back end of that data server is connected to host applications at the customer's remote site via a dedicated VLAN.

Alliance team member IBM's VLAN model represents years of experience, along with listening to customers and understanding their security needs. While VLANs are not unique, IBM's approach to implementing them is. We have separated our customers' Web and data servers on distinct VLANs to provide an added layer of security.

Focus on Redundancy and Protection Throughout the Hosting Center Environment

While IBM has developed the eBHC architecture to provide a scalable and flexible e-business hosting environment, we have not overlooked the importance of the physical structure itself. Redundancy and the protection it provides are evident throughout our Hosting Center facilities, as is the alliance's preparedness for any kind of outage or disaster.

The availability of the CCSAS CSE project's web hosting environment will be critical, affecting not only the Child Support Program's daily operations, but also its image. For this reason, IBM focused on continuous availability when building our Hosting Center facilities. Hosting Center facilities are designed to provide rapid recovery in the event of an equipment failure.

Dual Power Supplies

Our ability to provide the highest level of availability begins with connectivity to local power supplies. We offer dual power sources from different power substations with diverse routes to the building. These dual supplies enter the Hosting Center from opposite sides of the building.

In addition, the alliance provides redundant power distribution across the facility. In emergency situations, full battery/UPS backup power sources automatically take over. For extended power outages, the alliance activates 2000 KVA diesel generators that are brought into service by automatic throw-over switches. To help maintain sufficient fuel, the alliance also uses separate 15,000-gallon diesel tanks.

Water and Fire Protection

IBM Hosting Centers are equipped with fire and smoke detectors, alarms, and extinguishing systems consisting of Halon and water suppression technologies. Each Hosting Center has a wet pipe sprinkler system and under-floor leak detection system. For early detection, the alliance uses the incipient smoke detection system that is tied to both our fire alarm system and our under-floor leak detection system.

Dual Carrier Connectivity

The alliance's focus on availability continues with carrier connectivity. Dual telephone trunks from two different carriers enter the building from opposite sides. The alliance's redundant carrier connections include:

- Duplicate transmission facilities from distinct carrier switching offices
- Routers and switches
- Connectivity to primary and backup network control centers

Extensive Physical Security and Access Control

The alliance employs extensive physical security controls. Each alliance facility controls access using photo-ID badges, key locks, and security card reader systems. Access to every area of the building must be authorized before entry is permitted. In the Hosting Center's restricted access spaces, such as critical equipment areas, only specially authorized individuals may gain access. Video cameras monitor and record activity, supplementing an onsite security staff that is on duty 24 hours a day, 7 days-a-week.

Controlled Server Access

The alliance restricts access to the logical and physical server environment while providing the CCSAS project staff's designated server administrators with the access necessary to perform their work.

The alliance will implement procedures for resetting passwords for CCSAS CSE project users. The alliance will track CCSAS project staff and alliance individuals with password administration authority. When an employee leaves the project, their access to CSE environments will be removed. The alliance will use a checklist of de-activation steps to be performed when an employee leaves the project.

Coverage with Local and Centralized Server Administration

The alliance's Operations personnel perform the Hosting Center's administrative functions. These functions will be performed locally at the Hosting Center site that will supply the CCSAS CSE project's e-business hosting services, and are provided 24

hours-a-day, 7 days-a-week, 365 days-a-year to provide rapid assistance in solving problems and answering inquiries.

The Hosting Center site serving our e-business hosting operation will handle the procurement and physical installation of the CCSAS CSE servers. Once the hardware is installed, the *logical build*, which provides the operating system software, , systems management components, and basic network connectivity, will be managed centrally from the IBM West Service Delivery Center (SDC) in Boulder. . Monitoring the servers and the day-to-day operations will also be handled by the SDC. A central Configuration Database will be used to track and manage the placement of each server at the Hosting Center. The only ongoing requirements for local administration involve onsite support of the installed hardware and any required moves, additions, and changes to that hardware.

The alliance will produce Local Desk Procedures for System Administration (CDL TM 083) and Local Desk Procedures for Network Administration (CDL TM 084) for use by the alliance and CCSAS project system programmers and administrators supporting CCSAS.

The alliance's System Administrators provide ongoing server administration. A System Administrator specializing in that server type manages each server type (data, application, Web). Once the alliance's Design team has determined the logical build of each of the CSE system servers; the servers will be turned over to their respective alliance System Administrators. The alliance Systems Administrators will complete the configuration and implementation, and provide ongoing maintenance of the servers.

3C.14.1.2 Operations Problem Resolution

Operational problems go through several phases before they are fixed. Operational problems are time-critical and the faster the problem is identified, the faster it can be resolved. Our goal is to resolve problems before they affect CCSAS end users. First, the problem must be identified, which is accomplished by extensive server monitoring—*24 by 7 by 365* in our e-business Hosting Center. Once a problem is identified, the alliance will document, forward, and track it. To document an operations problem, the problem description, affected components, time, date, problem severity, originator and any other pertinent information is entered into the operational problem management system. The originator, usually an experienced operations professional, will make an initial evaluation of the problem and electronically forward the problem to the appropriate resolver group. Problems identified from any source, either reported to the help desk or uncovered through monitoring procedures, proceed through the problem management process identically. Possible resolver groups include server administration and support, and application/database support.

A key component of resolving time-critical problems is the assigning of a problem severity. A problem severity can range from one to four, with a severity one problem indicating that there is a major system outage. For severity one problems, we form a cross-functional team to focus on the problem. For severity one problems, alliance operations staff does not leave the operations center until it is resolved. Severity one problems get attention from the Service Delivery Manager, who has the ability to pull in

technical resources from our deep technical bench of over 3,000 hardware and system software specialists and system programmers in the alliance Service Delivery Center supporting the CCSAS project. As operational problems age their priority can be upgraded to increase the attention given to them by the production support staff.

Our technical staff will correct operations problems, and they will perform a root cause analysis to identify changes that need to be introduced into the CSE environments to reduce the likelihood of similar problems. Incident, Problem, and Issue Management will be performed in a manner consistent with those described in our Project and Technical Management approach. The goal of problem management is to reduce operational risk and increase the availability of the CCSAS system.

Operations Problem Management

Problem Management is an alliance System Management Control Discipline. As an SMC discipline, the alliance will approach problems and their resolution in a structured manner.

The alliance will use an Operations Problem Management Process for the CCSAS CSE project, as described in Section 3C.14.1.C below. Problem Management is the process of controlling and overcoming the problems that impact service delivery. At a high level, the process will include problem detection and resolution, and it will involve each step, procedure, and tool used to perform problem management. Project Management will include reporting and entering problems into a problem tracking facility, identifying recurring problems, and containing or reducing the impact and number of problems that occur.

The Operations Problem Management and the Problem Resolution Process are closely related. The operational problem management process and the problem resolution process, described in Section 3C.10, will exchange data to timely identify and resolve CSE problems. When a problem is identified by the operations staff to have an application-related component it will be manually communicated to a problem evaluator. A problem documented in the operations problem management system that includes both operational and application components will not be closed until it is confirmed closed in the application problem resolution process.

Problems should not be confused with issues. Issues, and the issues management process, are outlined in our Project Management Approach and deal with matters of concern at the project level. Operational problems are specific to the eight CSE operating environments. Operational problems by definition can only be corrected through a configuration change or some operator intervention with a malfunctioning component. If this criterion cannot be met then it should be reclassified as an application problem or an issue and be entered into the issue management process instead.

Operational problems can also be escalated to the project level issues management process if they cannot be resolved within the operational problem management process.

Finding Operational Problems Through Monitoring

Server monitoring is more than simply determining if a server is operational. To the alliance, server monitoring is a process of data collection that allows the alliance to verify the responsiveness of CSE System applications. The alliance's server monitoring functions provide those monitoring capabilities with an incomparable toolset, led by BMC Patrol. Tivoli and a wide array of performance and problem management tools enable the alliance to monitor the health and real-time status of the server environment. BMC Patrol also provides information on server performance trends.

Monitoring the Health of the Server Environment

Tivoli's suite of monitoring tools provides a comprehensive snapshot of the alliance-hosting environment at any point in time, enabling the alliance to proactively attend to performance issues and tune the operation. Tivoli provides a universal view of the hosting infrastructure and reports problems to a centralized location where they can be compared, filtered, and resolved without redundant resource allocation.

Tivoli will monitor each server in the e-business hosting environment, including each server's hardware, the processes running on the server, and the pathways to and from each server. When trouble is detected or an unacceptable threshold is reached, the Tivoli monitors generate an alert, known as an *event*. Events are reported to the Tivoli Enterprise/Console. The Console filters incoming events, consolidates them, drops duplicate or irrelevant events, closes self-correcting events, escalates severe events, and correlates cause and effect. Using rule-based logic to determine the severity and type of the event, the Tivoli Enterprise/Console will submit those events that warrant intervention to the alliance's problem management system, NPCA (Network Problem and Change Application). NPCA, in turn, will generate a trouble ticket for each event.

The trouble tickets entered into NPCA are immediately visible to the alliance's technical support staff onsite at the affected Hosting Center facilities and Network Operations Centers. The alliance's Help Desk is also notified through an interface to the problem management database. Alliance personnel who support hosting operations have real-time access to open trouble tickets, enabling them to track the status of high-priority problems and to quickly escalate specific trouble tickets.

The overwhelming majority of problems detected by the Tivoli monitors will be resolved before CCSAS end users become aware that a problem occurred.

Standard Tivoli Monitoring Services

Monitoring of the servers begins once a new server is installed in the Hosting Center. The standard monitoring tools are installed and monitoring is activated. The major Tivoli monitoring components are:

- Port Checking and Pattern Matching Monitor (PCPMM)
- TME 10 Distributed Monitoring

PCPMM monitors the network and application availability of servers by monitoring specific ports. PCPMM actively polls the Web servers and looks for patterns – such as a particular character in the first HTML tag – to determine whether Web sites are up and running. Status information is recorded in a log that is continuously checked by Tivoli's logfile adapter. When trouble is spotted, the logfile adapter forwards the associated messages to the Tivoli Enterprise/Console.

BMC Patrol monitors the applications, CPU, disk, memory, and processes for each server and platform in the Hosting Center environment. For example, for Windows NT servers, BMC Patrol provides application status information, as well as processor time, utilization information, and memory available information. For UNIX servers, BMC Patrol provides information about load averages, percentage of disk space used, and available swap space. As problem thresholds are reached for each of the monitoring applications, alerts are generated and submitted to the Tivoli Enterprise/Console and NPCA for trouble ticketing.

BMC Patrol is used for discovery, status monitoring, and fault management of the interfaces in the Hosting Center's IP devices. Those devices include the routers, switches, Web servers, database servers, load-balancing devices, and firewalls. NetView polls (pings) the interfaces at 5-minute intervals to check their availability. If no response is received from a poll, NetView generates a "trap" identifying the interface that is down and forwards an alert to the Tivoli Enterprise/Console.

The alliance will use performance-monitoring tools that are configured to assist in predicting system failures.

Server Resource Management

The alliance's BMC Patrol product reports historical trends of key server resources, such as CPU, memory, and disk utilization in support of users worldwide. BMC Patrol incorporates a series of supplemental programs and techniques that gather local server data, consolidate that data into DB2 for retention, and enable server utilization reporting directly from the Web. The alliance Service Delivery Managers can view the data gathered by BMC Patrol and discuss the results with users to identify issues that could affect availability and to plan for future growth.

The thousands of servers we currently manage through BMC Patrol include major vendor platforms such as IBM pSeries, zSeries, and xSeries eServers, and Windows NT, AIX, HP-UX, Sun Solaris, and Novell Netware. The alliance shall provide the to measure areas of system performance.

An Enterprise-Level View of Performance

BMC Patrol reports display an enterprise ("top down") view of the customer's server trends, in which hundreds of servers can be presented in a single view. Navigation down to individual server measurements is also possible. Report information is organized through a Red/Yellow/Green threshold methodology to quickly identify servers that

exceed key resource thresholds, saving countless hours of support time in analysis. In this way, our support is proactive rather than reactive.

Most resource management tools take a “bottoms up” approach, where costly volumes of data, reporting, and infrastructure are invested to build and present detailed server and application-level metrics, with no capability to show the enterprise view or to decipher the vast amount of information presented. BMC Patrol provides a common Web interface that offers a unified view across platforms and facilitates the acquisition of information.

Browsing capabilities provide the means to quickly and easily view daily, weekly, and monthly performance and capacity information. BMC Patrol can generate similar information on memory and disk utilization, and data availability, plus Red, Green, or Yellow action lists of servers that have reached performance or capacity thresholds. This wealth of information, which may be accessed according to selectable options, enables the IBM’s Service Delivery Center (SDC) personnel at the Hosting Centers to see what servers in the network may soon be out of capacity and how they should load balance or increase capacity to keep end-user service levels high.

The types of information shown on BMC Patrol screens are invaluable in helping to identify utilization problems, highlighting when (day, month, year, and time of day) a server starts to overload and the specific detail logs the SDC needs to review in order to find the reason for an over-utilization condition.

Performance and Capacity Planning Reports

The same information that BMC Patrol makes available for online viewing can be generated as reports. The alliance’s Performance Management and Capacity Planning team typically evaluate CPU, memory, and disk storage utilization reports. The CCSAS project staff can also view these reports.

The BMC Patrol server reports will display various daily, weekly, and monthly server resource trends covering server platforms. The BMC Patrol Capacity Summary Reports are platform-based reports. They provide both a graphical and tabular representation of data on CPU, memory, and disk status. These reports are monthly summaries and are meant to portray overall health of a given platform.

The combined CCSAS project team that is engaged in production support activities will use a report on system utilization. This report will include the Capacity Planning and Performance Management metrics used to evaluate the CSE environments.

Desktop Management

The alliance provides support for over one million desktop/server seats worldwide for commercial and government customers. As the information technology provider for the IBM Corporation, alliance member IBM Global Services has been supporting a distributed desktop and portable environment for well over a decade. In 1991, alliance member IBM began applying its extensive knowledge in information technology

management to the commercial and government marketplace. The alliance's approach to the distributed environment has been to apply the same robust processes, disciplines, and tools that we pioneered for the mainframe environment to the desktop.

Our desktop management solution for the CCSAS CSE project provides a set of processes, procedures, and automated tools for the management of the services required to support the distributed desktops and imaging systems. The desktop solution will be an enterprise approach of common desktop hardware and software platforms, configured, supplied and supported according to the three option approach described in Section 3C.13.4. As part of this strategic vision, the alliance will provide support of the desktop environment by managing and supplying:

- Desktop procurement
- Pre-delivery preparation and installation
- Contracted maintenance support
- Up to 1500 Installations/Moves/Adds/Changes per year
- Problem determination and resolution through a single-point-of-contact help desk, appropriate to the county's option
- Agent workstations that can double as image workstations to provide failover of image workstations
- An automated image restore facility on each desktop, that can restore corrupted software images from the workstation itself, without the need for deskside support or network connectivity
- Desktop refresh through the use of alliance resources and world-class partnerships in the areas of hardware/software products and skilled services. Desktop resources will be refreshed every three years for the duration of the contract

Desktop Operations

The alliance developed its Desktop Services to help customers integrate and manage the increasing complexity and cost of their multi-vendor client-server computing environments. The alliance has developed a customized desktop solution for the CCSAS CSE project to maintain and support the standardized desktop platform. To establish and maintain the desktop standards across the end users each county shall select to be configured according to one of the three options described in section 3C.13.4. . For Options 2 and 3 counties, backups will be performed using county backup procedures and equipment. For Option 1 counties extra disk capacity shall be provided on the image servers for backup of MS Office data files from the desktops

The alliance will provide automated distribution of software to the desktop through a centrally managed operation. The enterprise systems management tool set provides a means of managing and distributing software across multi-platform networks. It provides an efficient method for distributing, installing and controlling software across the network. This toolset will provide the CCSAS CSE project with a centralized software distribution solution, including the ability to update existing software with newer versions, and applying update patches or service PACs from a single console location.

Software distribution for desktop software fixes will be managed centrally, and distributed to the Image servers in each county, which will also function as software distribution servers. For option 1 sites, the desktops will be updated using a push approach from the county server. For option 2 & 3 sites, the county staff will be responsible for actually applying the updates by pulling from the county server. The Norton anti-virus software has the capability to schedule the software to automatically seek updated virus signature files via the Internet from the publisher's web site. Adobe Acrobat updates are also available via the Internet, and these updates will be made available in a coordinated manner to users according to the project's needs.

Incorporating Lessons Learned

Over years of delivering mission-critical projects for our customers, the alliance has learned the importance of a strong and comprehensive way of incorporated lessons learned into our operational processes. Our technical staff corrects operations problems, and performs a root cause analysis to identify changes that need to be introduced into the CSE environments to reduce the likelihood of similar problems reoccurring. These changes could include adjustments to operating system and system software configurations, CCSAS CSE application or database design and performance upgrades, or changes to the implementation processes and procedures. User training or other communication vehicles will be used, through our implementation teams at the LCSA sites, to reduce frequent end user problems that may impact system performance or availability.

The production support team will work with the implementation team in the development and review of CDL TM 039, the Post-Implementation Review Report. By reviewing the Post-Implementation Review Report, the production support staff may identify specific changes to the CSE environments that will reduce problems in future implementations. Once identified, these changes would flow through the standard operational change management process. We will incorporate lessons learned from CCSAS CSE implementation into the operations procedures included in CDL TM 075 System Operation Manual and CDL TM 076 Service Delivery Management Plan.

3C.14.1.3 Methods and Tools

The alliance's System Management Control Disciplines, which will be used to manage the CSE environments and provide the production support services, include the following component processes:

- Service Level Management
- Quality Control Management
- Operations Problem Management
- Operations Change Management
- Recovery Management
- Offline Processing Management
- On-line Processing Management
- Performance Management

The SMC Disciplines have been developed over 30 years during the management of complex computing environments. We use these disciplines on outsourcing engagements, which creates a consistency to our delivered services.

- Capacity Management
- Help Desk Management

SMC Disciplines, which are a total service offering, will enable the alliance to plan, organize, measure, and control the delivery of services. Through this series of processes and techniques, improved quality assurance and productivity will be achieved more systematically.

Figure 3C.14.1-1 A Total Service Offering - *The alliance's SMC Disciplines are an integrated set of processes and practices that create a reliable and high- performance environment.*

The knowledge and expertise of the alliance, as leaders in providing IT and telecommunications services, is brought to bear through the introduction of mutually-established methods and processes, and united service components to provide innovative technical solutions while achieving Service Levels.

The alliance will produce a System Operation Manual (CDL TM 075) and a Service Delivery Management Plan (CDL TM076) that is based upon the customized alliance SMC disciplines. The System Operation Manual will be based upon the J-STD-016-1995 J.2.3 with the addition of descriptions for the Help Desk and Customer Service Center. The Service Delivery Management Plan is very similar to the alliance's run book, which we develop for every hosting center customer.

SMC Discipline: Service Level Management

Service Level Management is the foundation on which delivery of Systems Management is built. In a cooperative effort, the alliance and the CCSAS project will develop a service level plan that will include:

- Develop the Service Level Attainment Report on a monthly basis that gathers and formats the information described in the Service Delivery Management Plan.Regular reviews of Service Level Attainment Reports with CCSAS project staff to determine how well commitments are being achieved over time
- Contract Management services and experienced support staff to monitor progress toward achieving contract commitments
- Service Level Management to facilitate obtaining Service Level Objectives through establishing specific, measurable goals, and through our administration of reporting procedures

Each of the other Systems Management Controls described below contributes to the Service Level Management.

SMC Discipline: Quality Control Management

Quality Control Management is an integral part of the alliance's SMC disciplines. Quality Control Management means the alliance will plan, organize, execute according to plan, measure, correct, and monitor the technical infrastructure Service Delivery Environments, which are:

- Hosting Center Operations
- Production Environment
- Production Simulation Environment
- Performance Environment
- Testing Environment
- Conversion Testing Environment
- Development Environment
- System Administration Support Environment
- Training Environment

SMC Discipline: Operations Problem Management

Operations Problem Management is the process of controlling and overcoming the problems that impact delivery of service. The alliance will provide an Operations Problem Management process that will include actions to take for problems that may impact systems service delivery. At a high level, the process will include problem detection and resolution, and it will involve each step, procedure, and tool used to perform that detection and resolution. Operations Problem Management will include reporting and entering problems into a problem-tracking facility, identifying recurring problems, and containing or reducing the impact and number of problems that occur.

The first step in Operations Problem Management is problem avoidance. The alliance will use Tivoli systems monitoring tools that will send automated alerts so that we will be aware of potential problems and, thereby, avoid many problems before they occur.

Some problems, however, will be unavoidable. In many such cases, the alliance's monitoring tools will provide alerts so that the alliance will be aware of problems even before they are discovered by the end-users, and the problems can be resolved before they can impact the services provided.

When a problem does occur, there will be several ways to identify it. Identifying the problem – whether it is discovered proactively through continuous automated monitoring or manually discovered by a user – will be the first step to recovery and resolution.

As part of the operations problem identification process, the problem description, affected components, time, date, problem severity, originator and any other pertinent information is entered into the operational problem management system. The originator, usually an experienced operations professional, will make an initial evaluation of the problem and electronically forward the problem to the appropriate resolver group. Problems identified from any source, either reported to the help desk or uncovered through monitoring procedures, proceed through the problem management process identically. Possible resolver groups include server administration and support, and application/database support.

A key component of resolving time-critical problems is the assigning of a problem severity. A problem severity can range from one to four, with a severity one problem indicating that there is a major system outage. For severity one problems, we form a cross-functional team to focus on the problem. For severity one problems, alliance operations staff does not leave the operations center until it is resolved. Severity one problems get attention from the Service Delivery Manager, who has the ability to pull in technical resources from our deep technical bench of over 3,000 hardware and system software specialists and system programmers in the alliance Service Delivery Center supporting the CCSAS CSE project. As operational problems age their priority can be upgraded to increase the attention given to them by the production support staff.

SMC Discipline: Operations Change Management

Operations Change Management is a consistent method for installing changes in such a way that the CSE systems continue to run efficiently, thereby reducing the possible negative impacts of change. Changes may result from problems reported via Problem Management, as a result of required hardware, software, or network maintenance, or to satisfy required application modifications. A change recommendation will be submitted by its author through the alliance's Operations Change Management system. The recommendation will be reviewed by the appropriate personnel, approved, and then scheduled. Operations Change Management will provide operational stability and availability while it assists the alliance to attain our Service Level Objectives.

Operational Change Management will involve risk analysis, planning, coordinating, monitoring, and communicating changes that affect CSE systems environment in accordance with ACF H-1. Operational Change Management will cover changes that can alter service delivery, including:

- **Hardware**: Installations, discountenances, and relocations of equipment in the Hosting Center, network, or CCSAS CSE sites. This will include Miscellaneous Equipment Specifications (MES) activity, and Engineering Changes (ECs).
- **Software**: Software changes including modifications to the system operating code, access methods, program products, or common support modules.

- **Application**: Application changes being phased into the production system. Application changes include alterations to processing programs, production, and the integration of new application functionality.
- **Procedure**: Changes to procedures that affect system availability or accessibility.
- **Environment**: Environmental changes involving the facilities associated with the Hosting Center and remote sites. Such environmental changes will include items such as air conditioning, chilled water, raised flooring, security, motor generators, and voice and data telecommunications.

SMC Discipline: Recovery Management

Recovery Management is the process of planning and establishing the recovery procedures that will be required to provide normal services in the event of a failure. Recovery Management will use the detailed documentation of the installed environment. Problems encountered in recovering the system will be entered and tracked through the alliance's Tivoli InfoMan tool. The intent will be to anticipate and contain the impact of systems resource failure through the use of documented procedures.

SMC Discipline: Offline Processing Management

Offline Processing Management is the process of controlling production offline work, including the scheduling of resources, processing of data and transactions, and the distribution of data and information between clients and facilities. The alliance's objective will be to achieve offline Service Level Objectives through an efficient use of resources.

SMC Discipline: On-line Processing Management

On-line Management is the process of coordinating the appropriate personnel, information, tools, and procedures required to manage on-line networks (both remote and local), operating systems, and applications. The alliance will use automated processes such as Automated Console Operations/Automated Network Operations (ACO/ANO) to monitor the application, the printing services, the network, and operating systems.

An alliance systems operator will be available to respond to each action, message, or response request generated by the system. The alliance operators will be equipped with extensive on-line and printed instructions for most situations. Each operator will use a workstation with window access to pertinent system operations and functions. Other operators in the Command Center will interconnect these workstations to a local area network, which will facilitate coverage across systems.

SMC Discipline: Performance Management

Performance Management is the process of planning, defining, measuring, analyzing, reporting, and tuning the performance of resources, including hardware, operating systems, applications, and services. The alliance will use our Performance Management SMC Discipline, described in detail in Section 3C.14.1.3.1, to analyze system performance.

SMC Discipline: Capacity Management

This process provides the alliance with the means to determine the systems capacity that will be needed to satisfy Service Level Objectives. The alliance will use our Capacity Planning SMC Discipline, described in detail in Section 3C.14.1.3.1, to maintain adequate system resources.

SMC Discipline: Help Desk Management

The alliance is comprised of leaders in the provision of Help Desk services. The alliance will use sophisticated automated tracking mechanisms to log, track, and monitor user calls so that calls and problems are not misplaced. Help Desk Representatives will be trained on CCSAS CSE applications to enable them to respond to each call and/or problem.

Detailed procedures will be written so that Help Desk Representatives know the precise steps to follow for each type of call. When appropriate, calls will be routed to other staffing areas to respond to questions and problems that cannot be resolved by the Help Desk Representatives.

The automated tracking system the alliance will use to log and monitor calls will provide regular reports including data for tracking numbers and types of calls and progress toward meeting Service Level Objectives. The alliance, in cooperation with designated representatives from the CCSAS project staff, will establish call categories, call priorities, and handling methods to meet the specific needs of the client.

Additional information on the alliance's help desk approach is located in Section 3C.14.3, Help Desk Services.

Security Procedures

Security cannot be just an afterthought; it must be designed into an e-business infrastructure. The key to security is to provide an integrated hardware, software, and network solution that is designed to work together and prevent the unauthorized disclosure of data. The alliance will provide the products, services, and expertise to secure the CCSAS CSE network and systems.

The alliance has extensive experience securing systems for our commercial customers, the federal government (IRS, Department of Defense), and state and local government customers. The protection of CCSAS CSE data is at the core of the alliance’s Security Service. Data Privacy, Security Procedures, Security Tools, Security Audits, and Physical Security are constructed around the CCSAS CSE data. Also, e-business security and encryption technologies will secure the physical transmission of data *to* and *from* the end users.

The alliance performs a detailed security screening of customer applications and data before any information resides at the eBHC. For CCSAS CSE, the security evaluation will include the IRS and Department of Justice requirements for data security. Our

security plan will include a role-based security approach to keep data protected and to allow disclosure only to authorized individuals or outside agencies. The security planning will identify CCSAS CSE's use of encryption, portable media procedures, physical security measures, processes for promptly removing security authorization from alliance and CCSAS project staff, and the control of the security authorization process.

IT Security as defined by the System Security Plan (CDL TM 078) is the process of providing the security protection of physical, logical, and network assets associated with delivery of services. It involves the definition of security strategies, policies, and procedures; managing their implementation by other Standard processes; and monitoring their compliance.

The System Security Plan includes the process and procedure for the following IT security components:

- Develop and maintain security strategies, policies and procedures
- Communicate security strategies, policies, and procedures across the e-business Hosting Center
- Develop security implementation plans to meet strategies, policies, and procedures
- Manage security implementation performed by other processes; for example:
- Maintaining physical access to controlled areas
- Implementing server security
- Providing supervisor-level access to servers
- Granting access rights to server data
- Verifying and revalidating access
- Resetting user passwords
- Monitor compliance to security strategies, policies, and procedures by other processes
- The alliance will conduct a risk analysis twice yearly and will perform special evaluations whenever a significant change to the system's physical security, hardware or operating system software occurs in accordance with ACF H-1c (CDL TM 078)
- Provide periodic reports as required
- Provide incident management for security breaches

The process begins with a joint alliance and customer definition of security strategies, policies, and procedures. The process described in the System Security Plan continues until the conclusion of the contract and these processes will be transitioned to the State at the conclusion of the contract

The System Security Plan will be used by the alliance production support staff to manage the CSE environment security issues and the CCSAS project staff to communicate and understand the security requirements of the data and systems to be developed and supported.

The alliance will conduct periodic risk analysis, which includes validating the appropriateness of the existing System Security Plan. The Security risk analysis is performed by an inspection of the eight CSE operating environments, intrusion log files, and the CCSAS CSE network. (CDL TM 079.

The alliance will create a System Security Report (CDL TM 081) in accordance with the System Security Plan and shall include incidents, impacts, causes and a description of new / revised controls to prevent reoccurrence.

3C.14.1.3.1 Analyzing Performance and Capacity Management

The alliance will use two of our SMC disciplines to analyze performance and to perform capacity management activities.

SMC Discipline: Performance Management

Performance Management is the process of planning, defining, measuring, analyzing, reporting, and tuning the performance of resources, including hardware, operating systems, applications, and services. Through Performance Management, the alliance will monitor, measure, analyze, tune, and report on system performance. Within the scope of this process, Performance Management will provide that the performance Service Levels Objectives will be achieved or surpassed for CSE environment systems by:

- Monitoring, analyzing, and reporting trends in system resource utilization and making tuning recommendations in order to employ the resources to their full potential and to avoid missing Service Level Objectives. Providing input to the Capacity Management process about resources required to meet present service levels and to plan for the future.
- If a performance problem is found, a problem management record will be created.

System Performance and Availability Report

A System Performance and Availability Validation Report will report on system performance and availability. The purpose of the Measurements and Reporting process is to enable reporting of the effectiveness and efficiency of service delivery. The alliance will produce a System Performance and Availability Validation Report on a monthly basis. The System Availability Validation Report will show the system performance metrics for the month that just completed, the prior month, and the same month for the prior year. The report format and content will be developed during the start of the project and provided to the CCSAS Project staff.

The System Performance and Availability Validation Report will be used by the combined CCSAS project team to monitor the effective performance of the CCSAS system. This report will be produced in accordance with the Performance and Capacity Management Plan (CDL TM 086) and will be analyzed by the combined CCSAS project team that is performing capacity planning and performance management.

Performance Management is an on-going process of measurement, analysis, and tuning to achieve the identified performance service level. Achieving Service Level Objectives will be the primary measurement of Performance Management's effectiveness. System-generated reports will compare the actual performance of the systems with the Service Level Objectives. These reports will be reviewed monthly with CCSAS project staff or more frequently, if performance problems exist.

Performance and Capacity Management Plan (CDL TM 086)

Capacity Planning, as described in the Performance and Capacity Management Plan, encompasses the process of planning for adequate IT resources required to fulfill current and future resource requirements. This process is limited to only providing capacity trends of operating system resources (CPU/Memory/Disk).

Performance Management, as defined in the Performance and Capacity Management Plan, is defined as the discipline of measuring, analyzing, reporting and tuning the performance of operating system resources (CPU/Memory/Disk).

The table below defines the differences between the Performance Management process and the Capacity Planning process.

Performance Management	***Capacity Planning***
Maximize performance of Resources	Maximize use of resources
Reactive	Proactive
Short term	Long term
Logical	Physical
Response Time	Business Planning / Financials
Existing Resources	New Resources

Figure 3C.14.1-2 Performance Management versus Capacity Planning – *Illustrating the differences between the Performance Management process and the Capacity Planning process.*

The Performance and Capacity Management plan will be used by the combined CCSAS project team that is engaged in production support activities. This plan will describe the Capacity Planning and Performance Management activities and processes to be used to manage the eight CSE environments.

Performance Management results are documented in the Performance and Capacity Management Plan. A System Utilization Report will be created based on the actual utilization captured. The Performance and Capacity Management Plan includes:

- Environments - Defines the CSE environments to be monitored and planned
- Performance and Capacity Plan objectives - The objectives of the Performance and Capacity Planning Process

- Scope of the Performance and Capacity Planning Process - The actual process and process diagrams for performance and capacity planning
- Dependencies - Dependencies that performance and capacity planning has on other production support areas
- Tools - Description of any tools to be used for performance and capacity planning
- Process measurements - Measurements to be used for performance and capacity planning
- Defects and exposures - Any known defects or exposures in the performance and capacity planning process, tools and techniques

SMC Discipline: Capacity Management

This process provides the alliance with the means to determine the systems capacity that will be needed to satisfy Service Level Objectives. Capacity Management will include forecasting the quantities of computer room facilities (electrical power, air conditioning, chilled water, raised floor area, etc.) to properly install the required systems resources. New hardware, if required and agreed upon by CCSAS project staff and the alliance, will be ordered and installed through the Operations Change Management process. Capacity Management will work to determine the *future* system capacity that will be needed to satisfy CCSAS CSE processing requirements. Capacity Management will involve collecting and reporting historical workload data, soliciting input from CCSAS Management to understand current and future capacity needs, and providing a capacity projection document for resources (processor, DASD, tape, print) that will be needed to achieve Service Level Objectives.

Capacity management plans and analysis are documented in the Performance and Capacity Management Plan.

3C.14.1.3.2 Disaster Recovery

As much as both the alliance and the CCSAS project staff would like to believe “it will never happen to us”, the truth of the matter is that it makes good business sense to prepare for disasters and to reasonably expend the resources that are required to protect our IT business investments.

A disaster is defined as a loss of the data center or central processing due to a catastrophic event or threat that would cause the vital processes to stop or become unavailable for an extended period of time. Examples of disasters are:

- Loss of a building or buildings to fire, flood, earthquake, or a hazardous material incident
- Regional or local damage or conditions from severe weather
- Other situations that cause catastrophic system outage

Disaster Recovery pertains to the execution of pre-defined and pre-tested disaster recovery plans and procedures that provides for off-site restoration of critical systems, applications, and IT business functions during the immediate post-disaster period. Disaster Recovery is implemented when the loss or delay, for an extended period of time, would:

- Result in significant loss of assets or revenue flow; and result in being unable to meet important customer commitments

The alliance, through IBM, has invested significant resources to provide protection of data center assets and facilities. The IBM San Jose e-business Hosting Center has modern monitoring, detection, and correction systems (fire, halon, power stability, physical security, water detection, environmental monitoring, and so forth) that will be used to reduce potential disaster exposures. The IBM San Jose Hosting Center will provide Uninterruptible Power Supply (UPS) and diesel generators to provide protection from disruptions of external electrical power supplies.

The alliance Recovery Management process and procedures will address recovery considerations for central site hardware outages. The IBM Recovery Management process will be tightly integrated with the Problem Management and Service Level Management SMC Disciplines to meet or exceed system and application availability objectives.

Disaster Recovery Offering

The alliance disaster recovery services will transfer the processing to the backup site, currently targeted to be located in Boulder Colorado within 48 hours of a declared disaster. A subset of the production environment (transaction and batch processing (without image support) and the ability to send out payments) will be restored in the event of a declared disaster. This includes 3 P690's and 1.5 TB of data. At the disaster recovery site the alliance will maintain the disaster recovery environment for CCSAS CSE in a state that will enable the 48-hour recovery for critical applications. During a declared disaster the system may perform in a degraded mode that does not meet production service levels.

The alliance will provide a 96 Hour Disaster Recovery solution for other production components including the image component and data warehousing, which includes one P690, up to 8 TB of image data on optical platters and 3 TB of additional data to support image processing.

During a declared disaster the system may perform in a degraded mode that does not meet production service levels.

The IBM Project Executive and Senior State Project Executive will mutually agree to make the declaration of a disaster.

The Disaster Recovery Plan will be used by the alliance operations personnel as a guide during the twice yearly disaster recovery testing.

Disaster Recovery Plan (CDL TM 090

The ability to recover a business from disaster begins with a well-documented Disaster Recovery Plan. Using the methods and tools developed over the past 30 years, the

alliance will assess business delivery, risk impact, vulnerability, interface to other business entities, personnel, and emergency preparedness as part of the Disaster Recovery planning process.

The scope of Recover from Disaster encompasses the protection, recovery planning, and testing of plans to recover assets and facilities to support critical business functions including, but not limited to, servers, minicomputers, mainframes, data networks, data, applications, skilled personnel and their required production environments.

In the event of a disaster, the alliance shall provide disaster recovery services in accordance with this Disaster Recovery Plan and assume responsibility for operating the equipment and providing services at the primary recovery facility.

The Disaster Recovery Plan will be used by the alliance operations personnel as a guide during the twice yearly disaster recovery testing. It will also be used in the event of a declared disaster to identify the systems to be recovered, the steps to be performed during that recovery and the roles and responsibilities of the alliance and the state.

The Disaster Recovery Plan will provide the plans and procedures for the restoration of the CCSAS CSE production environment and the network to connect the primary recovery center to the CCSAS end users. No disaster recovery services are provided for the local CCSAS CSE infrastructure in the 58 counties.

3C.14.1.3.3 Data Archive, Recovery, and Database Management

The alliance provides a comprehensive solution for disk subsystems management using IBM's Enterprise Storage Subsystem technology using commercially available backup media libraries and providing data archive and recovery support. The alliance will use dedicated space planners to implement and support the CCSAS CSE storage approach. The alliance will perform routine, periodic backups of child support enforcement system files, data files, application programs and documentation.

The alliance's storage approach includes:

- Identifying and maintaining the appropriate threshold levels to reduce potential space constraints
- Maintain a State-approved list of retention periods for application and operating system files and program versions in a Retention Control Document (CDL TM 089) in accordance with ACF H-5b.
- The alliance will store back-ups of the new statewide system files, programs, documentation, data files, etc. off-site in secure waterproof, earthquake-proof, and fireproof facilities in accordance with ACF H-5f.
- Communicating regularly regarding disk space availability, space-related failures, and areas of potential problems
- Creating and maintaining plans for incremental and full-volume backup and recovery

- Conduct regular differential backups of statewide system files, programs, documentation, data files, and DBMS database objects, which include full images and differential images at an appropriate interval, based on the specific database and data usage requirements;
- Conduct backups of the CCSAS CSE DBMS database objects without exclusion (e.g. user-defined items, related indexing information, etc.).
- The alliance shall retain off-site a four years plus 4 months history of the new statewide system database in a form retrievable through system recovery and restore procedures in accordance with ACF H-5e. This history shall be extended to seven years for IRS-related information
- Using the standard IBM LTO for tape backup operations
- Recommending efficient block sizes, migration criteria, and other parameters that affect disk usage
- Verifying that adequate file space is available for processing
- Conducting unattended backup and restore operations;
- Identifying possible enhancement and product opportunities for improved performance and notifying the CCSAS project staff of those opportunities
-

File Services

The alliance will manage files in a manner that will verify the availability and integrity of data. Included in such responsibilities, the alliance will:

- Verify that files under the alliance's control are current and available during requested access times.
- Initiate and complete required data processing activities to verify the data is processed, with data integrity for the processed files. This includes the ability to isolate portions of the physical database while bringing the rest of the database online for limited use.
- Conduct regular system and data backups and recovery procedures, (e.g., the ability to backup and restore interdependent databases and datasets), in a manner that would not impact scheduled operations. In general, alliance CCSAS CSE backup procedures will provide weekly full backup and daily incremental backups. Additional backups to support specific requirements, such as taking protective-backups prior to offline processing will be evaluated and employed where the alliance determines that they aid in recoverability, performance or other operational characteristics that are important to the CCSAS project. The alliance will provide recommendations to the CCSAS project staff regarding back-up and recovery considerations, such as improved levels of protection, efficiencies, and cost reductions.
- Recover or restore CCSAS CSE databases associated with the CSE environments to a consistent and usable state in the event of either a natural or technical disaster. Backups to support the disaster recovery process will be retained offsite and retrieved in the event of a declared disaster.
- According to procedures prepared by the alliance and reviewed with the CCSAS project staff, conduct routine monitoring and corrective action for intermediate files that are used for on-line processing and batch processing.

- Recover and restart systems operation after outages caused by system crashes (e.g., power outages) or server crashes.

Database Management

Database Administrators (DBAs) provide 24 X 7 X 365 coverage to address database problems and other situations that could impact service delivery. The alliance's Database Management support provides:

- Database implementation
- Database monitoring and tuning
- Physical database maintenance and recovery and the ability to isolate suspect portions of its physical database structure while bringing the rest of the database online for (limited) use, where possible by the finalized CCSAS CSE application and database design.
- The alliance will, on a regular basis, make backup copies of DBMS system tables such that this information is maintained in a form suitable for restoration
- Image copies
- Reorganizations
- Database tools evaluation
- Documentation of installed and supported databases
- Data administration:
- Maintaining and enforcing standards for database design, naming, and access
- Development assistance (including table generation and views)
- Problem determination and assistance with an application database
- File placement
- Disaster recovery for application databases

3C.14.1.4 Approach to Developing and Meeting Operational Service Level Objectives

Service Level Management is an alliance SMC Discipline. The CCSAS Executive Project Director and the alliance Project Executive have jointly established initial specific objectives consisting of those Service Level Objectives offered by the alliance. These Service Level Objectives may be modified later by agreement to include information assembled from a baseline of systems performance in Production. Those Service Level Objectives will be documented and will be supported by a series of reports. That Agreement will make Service Level Management possible by way of specific, measurable goals and by way of the reporting procedures that the CCSAS project and the alliance will administer.

The alliance and the CCSAS project staff will develop a Service Level Plan that will include:

- Regular reviews of Service Level Attainment Reports with the CCSAS project staff to determine how well commitments are being achieved over time
- Contract management services and experienced support staff to monitor progress toward achieving contract commitments

- Service level management to facilitate obtaining Service Level objectives through the establishment of specific measurable goals, and through our administration of reporting procedures

Service Level Management is supported by the other SMC Disciplines.

Service Level Recommendations

The alliance has combined its knowledge of the Child Support Systems, modern e-business systems and the known sizing information to estimate the CSE operational environments to achieve realistic Service Level Objectives.

3C.14.1.5 Approach to Transitioning Operational Responsibilities to the State

In providing a transition-of-operations activities for the CCSAS project, the alliance will develop an Operations Transition Plan that describes the tasks that must be executed to transition CCSAS CSE processing to another service provider or back to the CCSAS project staff or the State. The alliance uses the same transition-planning approach to move workload *in* or *out* of our facilities. The alliance will develop the Operations Transition Plan; including the identification of the transition time frame, the network configuration required for the transition period, the definition of transition roles and responsibilities, and the turnover of CCSAS CSE-specific processes.

Unique Transition Approach

The alliance's eBHC offers a unique approach to transitioning back to the State by offering a flexible range of services and being able to adjust the services provided over time. These services range from a co-location of processing hardware only or just the data center infrastructure, up to a fully managed solution including all of the components of the CCSAS CSE production support. As the CCSAS project staff gains knowledge and skills, specific operational responsibilities can be transitioned from alliance-delivered to State-delivered. Over time, at a pace decided by the CCSAS project staff, the services provided by the alliance will be reduced as the production responsibilities are assumed by the CCSAS project staff. We feel that this is a superior way of reducing the transition risk to both the CCSAS project staff and the alliance.

During each phase of the project the CCSAS CSE production support organization will maintain an on-site presence in the Project Management office. For production support activities the alliance will leverage the skills and efficiencies of our Service Delivery Center as well as the alliance staff in the PMO. The Service Delivery Manager will be permanently located in the PMO as well as the lead architect for production support services. Before the transition activities begin to accelerate, the production support organization in the PMO will grow to include additional staff, for example: Unix System Administrator, Network System Administrator, and a Database System Administrator. They will be located in the PMO during the transition period and they will perform three types of training activities:

- Classroom training where technical topics will be presented and the State will have the ability to ask questions to the alliance subject matter experts
- Activities where the alliance will be performing the production support activities and the CCSAS project staff will watch with the ability to ask questions
- Activities where the State will be performing the production support activities and the alliance staff will monitor and coach

The operational approach to transitioning services to the State will include providing following information and/or services to the Successor:

- Access to, and use of, hardware, software, personnel, third parties, and other resources then being used by the alliance and owned by the CCSAS project to provide the CCSAS CSE Services
- A Knowledge Management Plan to describe how the alliance will train the successor personnel and transfer knowledge on CCSAS CSE operations
- Such information regarding the Services in order for the Successor to assume responsibility for, and continue the performance of, the Services in an orderly manner, so as to reduce operational disruptions
- Copies of other information regarding the Services that are required to implement the Transition Plan, including:
- A list of libraries
- Copies of security tables and rules utilized in the provision of Services
- System modifications, including documentation, run sheets, and job scheduling
- Network documentation and diagrams, including modem configurations
- Copies of manuals required to enable the Successor to support the environment
- Identification and provision of the control release levels for Systems Software and copies of Operating Systems
- Freeze changes other than modifications necessary to address problems or to implement regulatory changes, unless otherwise directed by the CCSAS project staff
- Deliver to the CCSAS project staff technical specifications, database definitions, and materials and existing user documentation
- Deliver existing appropriate materials and information to enable the Successor to operate the Software, including job streams and associated job control language, run documentation, and applications support documentation
- Provide to the Successor reasonable access to the alliance personnel who were performing the Services
- Advise the Successor of current and pending project plans and status to enable the Successor to perform enhancement services with minimal disruption to CCSAS CSE operations

Upon transition back to the State or to another Service Provider, the alliance will provide transition information and services in an appropriate and professional manner.

Knowledge Management

Section 3C.7 describes the alliance approach to knowledge management. In that section we describe seven categories of production support staff that will support the CCSAS system after operational responsibilities are transferred to the state. These include:

- Database Support - Physical database support to manage database tablespaces, freespace and to plan, apply and test database package updates
- System Administrator - Monitors and maintains operating system and system software versions and updates
- Network Architect - Monitors and maintains network hardware and software versions and updates
- Infrastructure Architect - Monitors and maintains related service delivery infrastructure
- System Management Architect - Establishes and maintains the infrastructure to monitor system performance and capacity
- Security Architect - Performs security audits and monitors the security system to meet the information protection standards required by the CCSAS CSE project and all applicable government regulations protecting data
- Hardware Specialist - Maintains and upgrades the CCSAS CSE hardware
- Computer Support - Performs help desk and desktop computer support for the CSE environment

3C.14.1.6 Conclusion

The alliance will operate the CCSAS CSE system from our San Jose e-business Hosting Center. These hardened facilities will provide reliability that meets the 99.5% availability service level requirement. These facilities have 24 hour-a-day monitoring and are staffed with the alliance's dedicated professionals to supply CCSAS CSE services from *operations*, *monitoring*, and *technical support* to *system programming*.

3C.14.2 Maintenance

Proper system maintenance is required to deliver the levels of availability and reliability that are essential for the CCSAS CSE system. System maintenance is based on focused communication and a revolutionary system design that provides flexible access and is maintainable. Our production support activities are driven by adherence to proven system maintenance processes.

Maintenance is the cornerstone support for a well-designed, reliable application. It must include three important components:

The alliance will build maintainability into the system right from the start of the CCSAS CSE design process.

- A System Maintenance Plan, which will include the processes and procedures around maintenance activities
- System Maintenance Reports that will document the ongoing and expected system maintenance activities
- A Software Transition Plan (STrP), that will be used by the CCSAS project staff at the termination of the contract to transition the delivery of services to athe State.

3C.14.2.1 Overview of the Maintenance Strategy

System Maintenance is a process that starts well before production support. Maintainability will be designed, built, and tested into the CSE environment. Maintainability has been, and will continue to be, a major consideration of the alliance from the start of the design process, including application design choices and hardware and software selections, based on our collective experience building complex mission critical systems.

The System Maintenance Plan includes the detailed processes and procedures for:

Effective maintenance activities will allow DCSS and the LCSA to maintain a focus on their business.

- Application software fixes and enhancement work
- Database maintenance, including the installation of database environment upgrades
- System software maintenance for the software in each of the CSE environments from Production to Training
- System documentation maintenance for when system changes are made

The Change Control Process, which is an IBM System Management Controls (SMC) Discipline, ties system maintenance activities together and controls changes that are introduced into the system. Operations Change Control develops installation and de-installation plans for periodic application changes, system hardware and software changes and upgrades, environment or facility changes, and other changes that can affect service delivery.

The System Maintenance Plan will include the specific CCSAS CSE application maintenance processes and activities that must take place to support the application change and fix process. The alliance will use automated tools to track system changes and trace CCSAS CSE requirements.

Maintenance Scheduling

CCSAS CSE maintenance will include both scheduled and unscheduled maintenance for hardware maintenance, software maintenance and other maintenance activities to support database and other production support operations. Scheduled maintenance will occur in an agreed-to weekly maintenance window .During the scheduled maintenance window the alliance will perform hardware maintenance including preventive maintenance on a regular basis as suggested by the hardware manufacturer and also eBHC infrastructure maintenance. The alliance will coordinate eBHC and other maintenance windows, like application maintenance, and attempt to align maintenance windows when possible.

The alliance has taken significant precautions to minimize unscheduled maintenance. In the event of an unscheduled maintenance event, the alliance DPE will communicate the maintenance window to the CCSAS project staff and the State via email, pagers and telephone notification. The alliance will maintain a contact list of interested parties to be notified in the event of an unscheduled outage. The alliance will make every effort to minimize the disruption to CCSAS CSE operations by delaying maintenance activities to non-peak hours.

e-business Hosting Center Maintenance

Alliance team member, IBM, will provide support, routine and preventive maintenance, and agreed to upgrades for the hardware and software comprising the hosting infrastructure. This support will include coordinating support from third-party vendors that supply hardware and software to the hosting environment. The alliance will meet equipment and maintenance agreements. The alliance will manage this vendor relationship, providing a single point-of-contact for the CCSAS project, promoting compatibility of products, and enabling end-to-end support. The alliance will work closely with the CCSAS project staff in hardware planning.

The alliance will procure, install, maintain, upgrade, and support telecommunications software required for hosting connectivity. We will coordinate with local exchange and inter-exchange carriers to provide connectivity and maintain mutually agreed-upon performance standards.

The San Jose e-business Hosting Center will provide on-site hardware maintenance support. The alliance will schedule provisioning activities, and perform installation and maintenance activities in a regularly scheduled maintenance window approved by the CCSAS project staff. If we discover a problem that arises during a time that does *not* fall in the regularly scheduled maintenance window, we will notify the CCSAS project staff promptly if fixing that problem might adversely affect the availability of circuits and equipment. When a maintenance or upgrade activity necessitates system downtime, the alliance will notify the CCSAS project staff in accordance with the Maintenance Plan (CDL TM 091). The alliance will provide status reports on the progress of provisioning, installation, and maintenance activities.

In addition, the Service Delivery Team coordinates hardware recovery processes. Recovery scenarios are planned and rehearsed with the use of both online and printed procedures.

Using a highly efficient and accurate automated build-and-maintenance process, e-business Hosting Center personnel load and enable pre-designed software releases on CSE servers. Each software release includes a base operating system (BOS), monitoring tools, and the required monitoring agents. The releases incorporate the latest *fixes* and security patches, to help achieve compatibility with new servers and keep the configurations current with industry standards.

The following operating system and hardware configurations have been certified and are supported by the BOS service:

- Microsoft® XP, Windows NT® v4 or Windows® 2000 on selected Intel platforms
- IBM AIX® 4.3 and above on selected IBM pSeries (formerly RS/6000®) servers

The advantage of having the alliance provide system installation and management services includes:

- Providing a consistent and stable operating environment
- Enabling more efficient server management processes
- Helping to achieve application compatibility across servers
- Decreasing mean time for error recovery
- Leveraging alliance expertise and resources

3C.14.2.2 Application Maintenance

The primary objectives of Application Maintenance are to quickly respond to system problems, and to keep the CSE system current with policy, regulation, and requirements changes. The Application Maintenance team quickly responds to problems by managing the problem resolution and Level 3 Help desk functions. The Application Maintenance team keeps the CSE system current with changing requirements (produced from policy, legislation, or preference) by implementing enhancements to software functions. These three functions are described in the following paragraphs.

Problem Resolution

Our proposed Application Maintenance team will plan, develop, test, and prepare changes to address CCSAS CSE application problems. Application Maintenance will operate using the processes and controls in our Technical Management Approach. Section 3C.10 Problem Resolution is particularly relevant, as it describes our approach for addressing problems.

In summary, the primary Application Maintenance tasks related to problem resolution are to:

- Capture and log the requested change
- Review and analyze problem (includes scoping of the problem, including backward traceability to requirements, impact analysis, and prioritization)
- Receive and assign the work request
- Manage the problem backlog
- Fix and test problems
- Close out the problem
- Monitor and report on progress

The Application Maintenance team continues to execute configuration management processes during the problem resolution process.

Level 3 Help Desk

The Application Maintenance team serves as the Level 3 Help Desk. Level 3 Help Desk services are actually a subset of Problem Resolution services. As issues are unable to be addressed during Level 1 and 2 Help Desk support, the issue is logged as a problem. For Level 3 Help Desk, technical personnel – coders – research the problem by investigating the source code and application activity simulation. After researching the problem, it may either be classified as a problem that should be assigned to the problem resolution team (i.e., it really is a problem that will require programming change(s)) or the research will indicate that the software is performing as it should, and the problem is due to something outside of the software (e.g., hardware settings, improper user procedures, or incorrect user interpretation of the baselined requirements). For the latter, the problem will be assigned to the appropriate party for final resolution. Final resolution could take the form of a change request to the project scope change management process, if new requirements should be added.

Application Enhancements

Throughout the life of CCSAS CSE, changes will need to be accommodated. We expect changes to the application to be identified even before its initial implementation. The highly dynamic Child Support Enforcement environment practically guarantees frequent changes to the CCSAS CSE business rules. Policy changes are a fact of life as well in the Child Support Enforcement environment. Changes demanded by users, policy makers, legislators, and clients, as well as changing reporting requirements will require updates to the system. For application enhancements, our staffing plans are sized to

accommodate 1,500 function points each year for the two years after full system implementation.

Our Application Maintenance team will plan, develop, test, and prepare changes to the CSE system. This team will operate using the processes and controls in our Technical Management Approach and will involve CCSAS Project staff to plan and prioritize application maintenance activities. As such, the team operates under the same development tools, processes, and procedures used during initial application build – simply in a scaled down environment.

The application maintenance team, which reports into the Chief of Development, will be located in the Project Management Office. CCSAS project staff will be integrated by including team leads and application developers into the alliance application maintenance team, to be used to design, implement and test problems identified in the CCSAS CSE application and application enhancements.

System Maintenance Plan

The System Maintenance Plan (part of the Systems Operation Manual - CDL TM 075), defines the software and operational maintenance processes and procedures for the CSE system. The scope of the hardware and software to be covered by the System Maintenance Plan is the CSE application software running in the CCSAS CSE central processing site. Maintenance of the Help Desk and the Customer Service Support Center will be described in the System Maintenance Plan.

The System Maintenance Plan will be used by the alliance application development and maintenance organization, alliance production support, and the CCSAS project staff that is involved with the maintenance and production support of the system. This plan will identify the maintenance and operational procedures for CCSAS CSE application components.

System Maintenance Report

The System Maintenance Report will include a list of maintenance activities that were completed in the previous month and a list of those maintenance activities that will be planned for the next three months. The Operations Change Management Reports will be used to create the System Management Report.

The System Maintenance Report identifies the software and operational maintenance processes and procedures for CCSAS CSE that have been performed and the planned operational maintenance activities planned for the next quarter. The scope of the hardware and software to be covered by the System Maintenance Report is the CSE application software running in the central processing site. Maintenance of the Help Desk and the Customer Service Support Center will be included in the System Maintenance Report.

The System Maintenance Report will be used by the alliance application development and maintenance organization, alliance production support, and the CCSAS project staff

that is involved with the maintenance and production support of the system. This report will identify the maintenance and operational activities for CSE application components.

3C.14.2.3 Hardware and Software Refresh Cycle

The alliance is well positioned to keep the CCSAS CSE infrastructure, systems, and applications current with industry advances. The alliance will provide on-going assessment of new technologies and recommend those products that will enhance CCSAS CSE operations. We will look at best practices, in combination with new technologies, to recommend technology and products that will support the CCSAS project's goals and initiatives.

Using the knowledge gained through both Research and Development investments and service delivery to our worldwide customer base, the alliance will recommend implementing proven processes and methodologies to maintain CCSAS CSE technological currency in its IT and telecommunications infrastructure, systems, and applications.

The alliance suggests an initial refresh schedule that is based on our knowledge and experience with projects like CCSAS. The technology refresh schedule we recommend is as follows:

- Desktop Technology – 3 years
- Storage Technology - 5 Years
- xSeries, pSeries Server Technology – Refresh requires a change request

Long-Term Support Strategy

The alliance's long-term technical refresh and support strategy is to evaluate price/performance and maintenance costs of currently deployed technologies on an annual basis and to refresh those technologies at a minimum on the timelines identified above.

Our technical solution will utilize the IBM pseries 690 as the primary Unix platform. These systems have the ability to logically partition the system into smaller subsystems that will allow the operation of multiple CCSAS CSE functions on a single platform. The size characteristic of these servers, which is generally large in scale, reduces hardware maintenance and enables dynamic allocation of resources to meet changing needs. This refresh and maintenance strategy is more easily and quickly accomplished than the alternative of deploying a larger number of smaller servers. The pseries 690 also compares very favorably in terms of mean time to failure due to the memory check and fault characteristics built into the system. This system detects failures that are about to happen and takes corrective action to maintain operation, where possible. The maintainability characteristics of the pseries 690 is:

- Copper and SOI technology - Improve processor performance and reliability while using less power and producing less heat, thus conserving energy otherwise required for operation and cooling.

- Chipkill memory - Designed to reduce system outages, caused by memory failures and minimizes the potential for loss of data.
- Logical Partitioning - Permits multiple applications to be consolidated on a single server, reducing the number of systems to manage and maintain. Offers greater flexibility in using available capacity and matching resources to changing business requirements.
- Redundant hot-plug power and cooling subsystems - Enhance system availability as cooling fans or power supplies can be changed without interrupting operations.
- Built-in service processor - Designed to continuously monitor system operations and take preventive or corrective actions for quick problem resolution and high system availability. Allows diagnostics and maintenance to be performed remotely.
- Dynamic processor, memory, L2/L3 cache and PCI bus deallocation - Capable of automatically deallocating resources when impending failure is detected, thus enabling applications to run without interruption.

3C.14.2.4 Approach to Transitioning Maintenance Responsibilities to the State

In providing a Software Transition Plan, the alliance will describe the tasks that will have to be done to transition CCSAS CSE maintenance from the alliance to another service provider or back to the State. The alliance uses the same transition planning approach to move workload *in* or *out* of our facilities. The alliance would work with the CCSAS project staff to develop the transition plan, including the identification of the transition time frame, the definition of transition roles and responsibilities, and the turnover of CCSAS CSE-specific processes. Because of the unique nature of transitions, the alliance will estimate the transition effort upon notification of the need to transition and the scope of the transition.

During each phase of the project the CCSAS CSE production support organization will maintain an on-site presence in the Project Management office. For production support activities the alliance will leverage the skills and efficiencies of our Service Delivery Center as well as the alliance staff in the PMO. The DPE will be permanently located in the PMO as well as the lead architect for production support services. Before the transition activities begin to accelerate, the production support organization in the PMO will grow to include a Unix System Administrator, Network System Administrator, and a Database System Administrator. They will be located in the PMO full-time during the transition period and they will perform three types of training activities:

- Classroom training where technical topics will be presented and the State will have the ability to ask questions to the alliance subject matter experts
- Activities where the alliance will be performing the production support activities and the CCSAS project staff will watch with the ability to ask questions
- Activities where the State will be performing the production support activities and the alliance staff will monitor and coach

The Software Transition Plan (CDL TM 093) will provide the following information and/or services to the Successor:

- Access to, and use of, hardware, software, personnel, third parties, and other resources then being used by the alliance to provide the CCSAS CSE Services
- Information regarding the Services in order for the Successor to assume responsibility for, and continue the performance of, the Services in an orderly manner, so as to reduce operational disruptions
- Copies of the following other information regarding the Services that are required to implement the Transition Plan:
- Documentation of existing maintenance process and procedures
- A list of CCSAS CSE hardware and software in use at the San Jose e-business Hosting Center
- A list of scheduled maintenance activities planned for the CSE environments
- Identification and provision of the control release levels for the application's systems software, as well as copies of Operating Systems
- Reasonable access to the alliance personnel who were performing the Services
- Advice on the current and pending Project Plans and status to enable the Successor to perform enhancement services with contained disruption to operations
-

3C.14.2.5 Conclusion

Maintenance activities fit well into the alliance's standard procedures (SMC Disciplines) for maintaining and operating both the software systems and the managed environments to support the CCSAS project. Our approach to designing and building the CCSAS CSE system takes into account the ease of maintenance and the impact that simple and effective maintenance has on our ability to provide a reliable and responsive system.

3.C.14.3 Help Desk

The alliance currently provides Help Desk services to over 1.5 million customers. Our Customer Support Representatives use knowledge-based searchable databases to resolve known problems quickly. We also offer Web-based self-help facilities for those end users that prefer that medium. No matter which venue an end user chooses, we will use our time-tested Help Desk processes to meet the needs of CCSAS users.

The alliance will implement and provide a single point-of-contact (SPOC) Help Desk and project-wide Problem Management system for CCSAS users. Co-locating the CCSAS CSE Help Desk with the alliance Service Delivery Center in Boulder Colorado will enable the CCSAS CSE Help Desk to deliver more efficient services by leveraging the infrastructure, skills and staffing of the Service Delivery Center. The Help Desk infrastructure can readily adapt to meet the growth of the CCSAS user base over time. Prior to transition back to the State the alliance will move the Help Desk to the alliance Project Management Office to facilitate training activities prior to State takeover of Help Desk Services.

3.C.14.3.1 Overview of the Help Desk Functions, Processes, and Procedures

The alliance provides Help Desk services for over 1.5 million end users.

The CCSAS CSE Help Desk will provide Level 1 support for in-coming calls by answering the call, capturing call information, assigning a problem severity, and tracking the call to resolution. Calls to the Level 1 Help Desk may be answered through automation, but will not solely be automated. Level 1 calls are answered by the Interactive Voice Response Unit and then routed to an appropriate Level 1 Help Desk Representative. The alliance will use an automated Help Desk tool for problem tracking. Level 1 support is defined as the initial person who takes the incoming customer call, opens a problem ticket and provides initial problem determination that frequently results in problem resolution on the initial call. Level 1 support will provide problem determination support to CCSAS users. The following are features of the CCSAS CSE Help Desk:

- The CCSAS CSE Help Desk will answer Help Desk calls during the hours specified in SOW TM 14.3.4.
- Help Desk Services will support the CSE environments
- Ninety percent of the calls will be answered within 35 seconds

- Help Desk reporting, including feedback regarding problem resolution, open and closed problem accounting, and Help Desk Service Level Attainment reports
- Verification of problem resolution with the initiator before closing the problem record
- The CCSAS CSE Level 1 Help Desk, which will accept each help desk inquiry, will be located in Boulder Colorado to leverage alliance resources to support high-volume call periods
- The CCSAS CSE Level 2 Help Desk support will be provided from several locations, including:
    - Level 2 application support will be provided by Call Center staff trained in the CCSAS CSE application at the Service Delivery Center
    - Level 2 desktop and operations support will be provided by the Service Delivery Center in Boulder Colorado
- The CCSAS CSE Level 3 Help Desk support will be provided from several locations, including:
    - Level 3 application support will be provided by the alliance in the Project Management Office in Sacramento
    - Level 3 operations support will be provided by staff in the Project Management Office in Sacramento and the Service Delivery Center in Boulder Colorado

The CCSAS CSE Help Desk has been designed and sized to handle 10,400 users, at an average rate of 0.7 calls per user per month, for an anticipated maximum of 7,280 calls per month. The average call duration is estimated to be four minutes

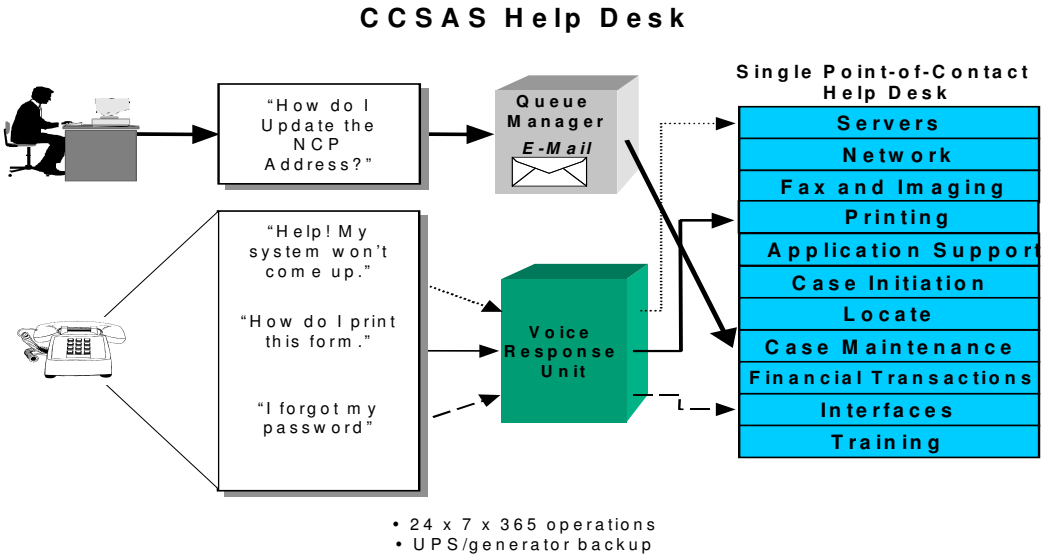


Figure 3C.14.3-1 CCSAS CSE Help Desk - *The alliance's Help Desk will provide problem resolution for CCSAS CSE end users.*

An experienced team that has become an industry model for similar technical support groups consistently administers our support services, shown in Figure 3C.14.3-1. In fact, the alliance is routinely called upon to assist companies around the world in setting up and managing their Customer Service Help Desks.

Our Help Desk organization offers the CCSAS project highly-available Help Desk support, performance reporting, end-to-end problem management, server management, vendor coordination, and consulting services through a single point-of-contact. We provide CCSAS users with access to the vast resources of the alliance, which simplifies problem reporting, achieves prompt problem resolution, and provides continuous feedback.

Fully integrated into our Hosting Centers, our support facilities deliver the same problem management services and in-depth technical support no matter where the user is located or the method he or she uses to report problems. While most Web-hosting providers offer server management, few have the resources in place to deliver the alliance's level of comprehensive support.

Help Desk Lead

The Help Desk Lead reports directly to the Service Delivery Manager. The Help Desk lead will define help desk operating policies and provide personnel/administration support, such as *scheduling*, *performance evaluation*, *personnel management*, and *staffing* and is responsible for developing the Help Desk Plan and monthly Help Desk measurements and reports. As the owner of Help Desk transition and Help Desk operations, the Help Desk Lead will be onsite at the CCSAS CSE Help Desk location to manage Help Desk staff to meet peak call volumes. The Help Desk Lead follows the resolution of customer satisfaction issues, plans meetings to communicate Help Desk issues to the CCSAS project staff, and develops prevention/improvement plans.

The Help Desk Lead will be responsible for incorporating State staff into the Help Desk services. The State staff can be used as Help Desk Representatives at any level of the Help Desk Support structure.

County and State Help Desks

The CCSAS CSE Help Desk will operate in conjunction with State and County provided help desks. As a result of the three option approach for desktop / LAN, option 2 and 3 counties are required to provide a Level 1 help desk, which will serve as the first point of contact. Also, to support the WAN and associated components, HHSDC are required to provide Level 2 help desk. The following chart summarizes the division of help desk responsibility:

Problem	Option 1	Option 2	Option 3
1. Questions about CSE Application (IVR & FAQs, then help desk).	alliance	alliance	alliance
2. Problems with CSE Application (e.g. error messages)	alliance	alliance	alliance
3. Desktop or desktop apps problems / administration	alliance	County	County
4. E-mail problems / administration	alliance	County	County
5. Desktop Break fix	alliance	alliance	alliance
6. Password administration - Application	alliance	alliance	alliance
7. Password administration - LAN, Desktop	alliance	County	County
8. LAN infrastructure administration (e.g. DHCP problems, moves / adds, connectivity issues)	alliance	alliance	County
9. LAN break / fix	alliance	alliance	County
10. WAN (e.g. site router, cable from router to switch, circuit, WAN, eBHC router)	HHSDC	HHSDC	HHSDC
11. Server environment	alliance	alliance	alliance

Figure 3C.14.3-1 Help Desk Responsibilities – *The table identifies responsibilities for help desk depending on the desktop option chosen.*

Level 1 Help Desk Services

The Level 1 Help Desk is the first point of contact for an option 1 user. For option 2 & 3 counties, the county's Help Desk shall transfer calls to the alliance Help Desk for problems identified as alliance responsibilities in Figure 3C.14.3 -1.

Level 1 calls are answered by the Interactive Voice Response Unit. If the problem cannot be resolved by the IVR, the call is routed to an appropriate Level 1 Help Desk Representative. The Level 1 Help Desk Representative, using the automated Help Desk tool, opens a problem ticket that shall be used to track and manage the problem until it is closed. The Level 1 Help Desk Representative then performs initial problem determination that frequently results in problem resolution on the initial call.

Level 2 Help Desk Services

Technical specialists provide level 2 services with specific knowledge in their area of expertise.

The Level 2 Help Desk will have resources and specialized skills to assist with the resolution and closure of problems. . Level 2 support organizations include the e-business Hosting Center for production support issues involving the server, the Application Support staff for questions concerning the CCSAS CSE application and the CCSAS business processes, and the State's HHSDC Help Desk for problems with the Wide Area Network.

Alliance Level 2 Help Desk Services

The alliance's Level 2 Help Desk provides the technicians who have the technical experience and support they need to resolve application, hosting and server-related problems.

The Level 2 Help Desk shall operate as an extension of the Level 1 Help Desk. The alliance's Level 1 Help Desk will be in direct contact with the Level 2 Help Desk technicians who will know the CCSAS CSE host configuration and will be trained to provide efficient service. A Level 2 Help Desk technician will be contacted by the Level 1 representative, either via an automated transfer or a call to the appropriate Level 2 Help Desk. The Level 2 technician shall then call back the originating user if necessary. . While the problem is active, it will be the Level 2 technician's job to keep the originator of the problem apprised of its status at regular intervals, based on the problem's severity as defined in the Help Desk plan (CDL TM 094). The Level 2 technician will also make updates to the problem ticket at agreed-upon intervals. Upon resolution of the problem, the Level 2 technician closes the original ticket, either through automated means, or by calling back to Level 1.

Until the problem is resolved, the alliance Level 2 technician will escalate the problem to the appropriate personnel, as needed. The alliance specialists and engineers who are called in to resolve the problem will continue to work toward the problem's resolution. If necessary, they too can escalate the problem—either *internally* to the appropriate technical support group or, for specialized help, to the alliance's Server Assistance Program (SAP) or *externally* to the appropriate third-party hardware or software manufacturer. In that case, the alliance specialist or engineer will monitor the vendor's

performance and report status information to the Level 2 technician who is charged with overseeing the resolution.

For the CCSAS project, this transfer of care from a Level 2 Help Desk technician to a specialist will be transparent. The Level 2 technician will maintain ownership of the problem(s) and will continue to be the focal point for the communication with the person who reported the problem. The DCSS and the LCSA project staff will need not explain and *re-explain* the same problem to a series of technical experts who were called in to resolve it. The alliance's Level 2 Help Desk personnel will provide this service and will maintain contact with the CCSAS CSE Help Desk technicians, simplifying the CCSAS project staff's role in the problem-solving process with a single interface for reporting and tracking problems.

HHSDC Level 2 Help Desk Services

The State shall provide the Level 2 Help Desk services for problems associated with the Wide Area Network. This includes the WAN itself, the connecting 'tail circuits', the routers and CSU / DSU. As part of this service, the State shall:

- Proactively monitor the network (to the routers)
- Monitor tail circuits in case of failures (using dial up to router to view circuit)
- Provide proactive notification to CSE Help Desk and County Help Desks of network problems.
- Accept transfer of responsibility from the CSE Level 1 Help Desk for WAN problems.
- Administer, service and fix the assets described above.
- Monitor and manage problem tickets opened by the CSE Level 1 Help Desk.
- Communicate resolution of problems and problem tickets to the CSE Level 1 Help Desk and the County Help Desks using the problem ticket identifiers provided by the CSE Level 1 Help Desk.

Level 3 Help Desk Services

The Help Desk will have Level 3 resources to assist with the resolution and closure of problems. These Level 3 resources will have highly specialized skills. Level 3 support organizations include the e-business Hosting Center and Application Support staff to address problems with the application or the system hardware and resources, and extend to product organizations such as WebSphere or DB2 support.

Customer Support and Operations Change Management Tools

The alliance will use an automated tool set to help resolve problems and monitor Help Desk performance in real time.

The alliance will use an automated Help Desk tool set that is integrated into the CCSAS CSE Operations Problem Management and Operations Change Management services. This tool set is comprised of Help Desk systems, like IBM's help desk tool called eESM, ManageNow and HelpNow. The eESM application is the core for the tool set; it provides a Help Desk tracking mechanism with access control mechanisms, searchable database of CCSAS CSE customer service scripts, problem

reporting, Help Desk Service Level reporting, and a means for tracking problems through a closed- loop customer support process. The alliance will reuse existing Help Desk scripts for common CCSAS CSE components like the desktop, Windows operating system and the browser on the desktop wherever possible.

The Help Desk Representatives, who will provide Level 1and Level 2 support, will access the eESM database through an application that will run on their workstation. ManageNow and HelpNow provide access to the single problem-tracking database designed for Level 2 and Level 3 technical staff for problem resolution, enabling the alliance to deliver a closed-loop process for customer support.

Our automated tool set provides a Notification/SLA/Escalation/Paging server that is used to provide automated paging services to the support staff when certain capacity thresholds are reached or when there is a system outage. This feature allows each server to have a predetermined escalation chain. The notification process, for example, works in the following way: the Tivoli Management Enterprise (TME) tools alert the eESM message/alert board which, in turn, alerts the Notification server. The TME infrastructure, which includes Tivoli Enterprise Console, Netview, and Distributed Monitoring dedicated in the e-business Hosting Center will monitor the CSE environments and send alerts directly to the CCSAS Help Desk. Tying together the management tools and Help Desk will assist the alliance to identify and resolve problems faster.

Help Desk Plan (CDL TM 094, TM 14.3.1)

The alliance will develop a Help Desk Plan. This plan will include the details required to establish the Help Desk, build the Help Desk infrastructure (including the VRU system), and train the Help Desk Representatives. Help Desk Representatives will have training so that they have knowledge of the CSE application, which will be included in the Help Desk Plan.

The Help Desk Plan will be documented using CCSAS standard office tools. It will include task durations and dependencies, and it will be integrated with the other Project Plans designed for the CCSAS project. The Help Desk Plan will include:

- Identification of the Call and Incident processes
- Description of the integration of Level 2 and Level 3 resolver groups and the integration with other local help desks
- Service level management and reporting
- Description of the help desk planning for the implementation of the CCSAS CSE Help Desk
- The Staffing Plan and acquiring Help Desk staff
- Communicating Help Desk capabilities as part of the rollout of the CSE Application
- Establishing Help Desk scripts
- Formulate interfaces to Level 2 support staff
- Methods for monitoring Help Desk operations
- Reporting Help Desk metrics

- Establishing a Help Desk Web site, including Frequently Asked Questions (FAQs) that is available to DCSS, LCSA and FTB via the CCSAS CSE network. (TM 14.3.8)
- Designing and conducting Customer Satisfaction Surveys

Help Desk Processes

Alliance team member, IBM uses the same, proven Help Desk processes and procedures for every customer.

The alliance will use sophisticated automated tracking mechanisms to log, track, and monitor user calls so that calls and problems are not misplaced. This Help Desk Process, described in Figure 3C.14.3-2, shows the alliance's closed-loop method. Help Desk representatives will be trained on the CSE application to enable them to respond quickly and efficiently to each call and/or problem.

Detailed procedures will be written so that Help Desk representatives will know the precise steps to follow for each type of call. When appropriate, calls will be routed to other staffing areas to respond to questions and problems that cannot be resolved by the Level 1 Help Desk representative.

The alliance will use an automated tracking system to log and monitor calls; it will provide regular reports for the CCSAS project - tracking the number of calls, the types of calls, and the progress toward meeting the mutually-negotiated Service Level Objectives. The alliance, in cooperation with designated representatives from the State, will establish categories of calls, call priorities, and handling methods to meet the specific needs of the Department of Child Support Services.

The alliance will provide access to the Help Desk tracking tool to the CCSAS project staff as described in TM 14.3.3. and TM 14.3.7.

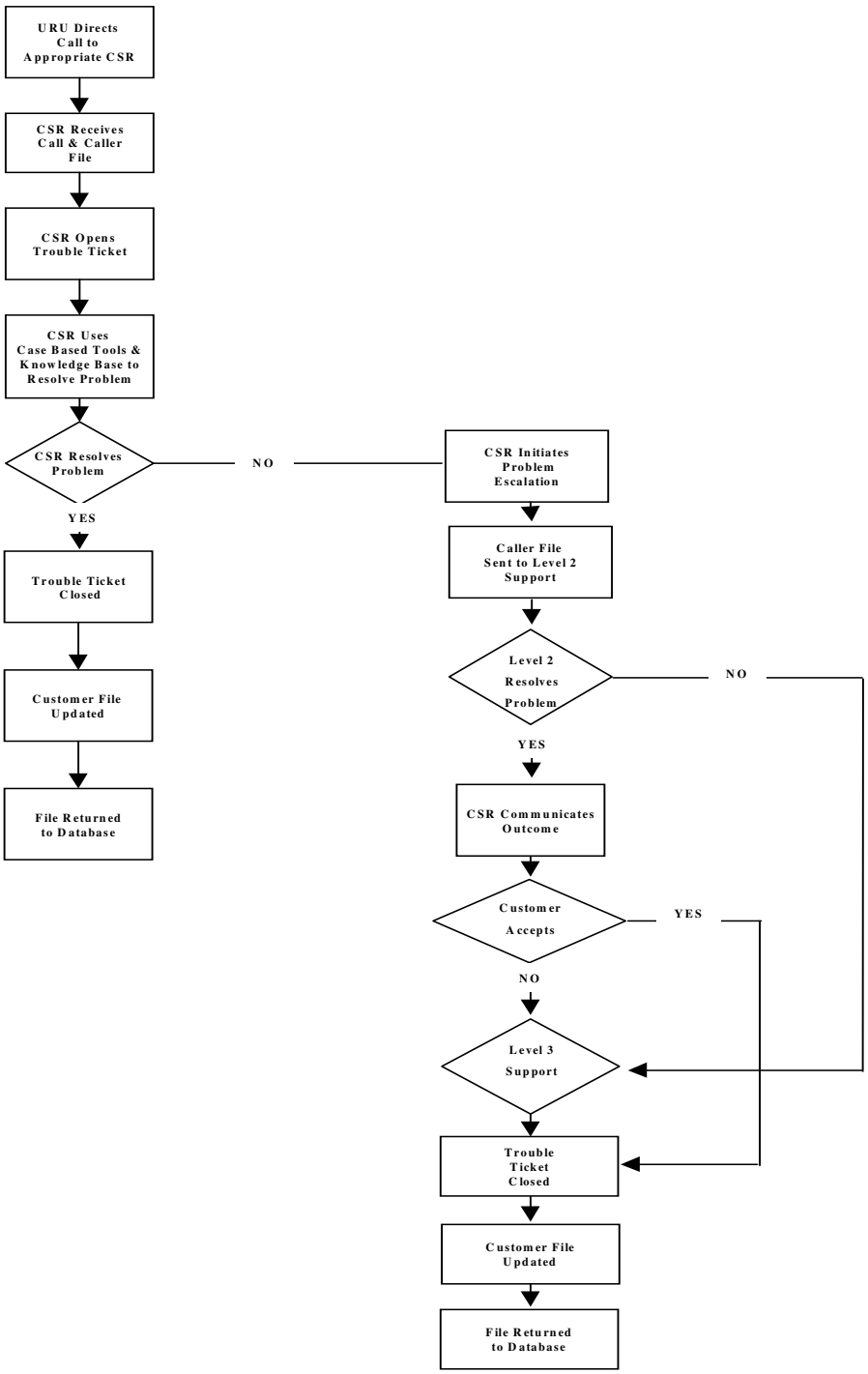


Figure 3C.14.3-2 Help Desk Process - *The alliance has a closed-loop process for Help Desk Trouble Tickets.*

Automated Call Distribution System

The alliance Help Desk will employ Automatic Call Distribution (ACD) and Interactive Voice Response (IVR) technology, which will effectively distribute incoming calls to the Help Desk Representative best equipped to respond to the caller's needs. Further, the Help Desk will employ Computer Telephony Integration (CTI), which will link telephone and computer systems to increase functionality and efficiency. CTI technology enables agents to provide intelligent support by calling up stored information about the caller that resides in a Call Center database, and directs it to the Help Desk Representative's computer screen. The alliance will provide training on the automated call directing system and problem tracking system to the CCSAS project staff as specified in TM 14.3.6.

The ACD system will create reports, as specified in TM 14.3.5.

Help Desk Staffing

The Help Desk will be adequately staffed with trained professionals to meet the needs of the CCSAS users. Initial staffing will be based on a projected call volume. Adjustments to staffing will be made as the alliance assumes responsibility for this function and begins to analyze both the volume and the nature of calls coming into the Help Desk. Initial staffing will be made up of a Help Desk Lead (who will have responsibility for customer relations, customer satisfaction, queue management, crisis monitoring, and support staff training), Technical Leads, Team Leads, and Customer Service Representatives. A sample Help Desk staffing might have the following positions:

- **Help Desk Lead** - defines operating policies and provides personnel/administration support, such as *scheduling*, *performance evaluation*, *personnel management*, and *staffing*;
- **Customer Satisfaction Manager** - conducts Customer Surveys to build a consistently high level of satisfaction. The Customer Satisfaction Manager communicates to the Help Desk Lead and Customer Relations Manager any patterns of dissatisfaction that require attention. He or she documents survey results in the Customer Satisfaction database
- **Training Coordinator** - maintains inventory and identifies training requirements. Develops training requirement specifications for trainers (within and outside the Help Desk)
- **Queue Manager** - monitors the queue or queues assigned to validate that each request is assigned an owner and is responded to within the appropriate time frame, based on contractual requirements
- **Level 2/3 Manager** - A Level 2/3 manager is responsible for establishing and maintaining a Level 2 or Level 3 service area for the CCSAS project staff. Level 2 and Level 3 groups resolve those problems that require more detailed knowledge than Level 1

The alliance Help Desk Customer Service Representative uses standard, closed-loop processes for not only tracking and resolving customer calls but also for making sure that the problem's root cause is identified and corrected. Level 1 Customer Support will be the end user's sole point-of-contact. Calls that are beyond the capabilities of Level 1

staff will be directed to Level 2 technical experts. The alliance Level 2 technical experts will be available at the Help Desk to provide technical support to users.

Help Desk Integration with the Problem Resolution Process

The Help Desk is integrated with other production support processes through our SMC disciplines. There are two primary considerations for the integration of production support services, they are the quality of information available to Help Desk Representatives to assist them identify operational problems and the rapid movement of problems reported to the Help Desk into the Operations Problem Management process.

To provide a high level of service, the CCSAS CSE Help Desk must be aware of operational problems to simplify problem determination and to effectively communicate with CCSAS project staff, DCSS users and LCSA users. Each Help Desk Representative will have access to current alert and outage information.

Help Desk tickets that are routed to Level 2 and Level 3 resolver groups will be quickly evaluated by the production support team. In many cases the Level 3 support person will also have responsibility to resolve the operational problems. Upon confirmation that the Help Desk ticket is an operational problem, the problem will be logged into the operations problem management system and the problem will be documented. From this point, the resolution and escalation process (including any integration with the PMP Problem Resolution Process) will be identical for all problems regardless of their source. A detailed description of the Operations Problem Management and the integration with the PMP Problem Resolution Process is described in Section 3C.14.1.B. As a final step, problems reported from the CCSAS CSE Help Desk that aren't closed on the by the Level 1 agent are closed after a callback to the end user verifies that the problem is fixed.

The Help Desk will respond to Project Staff, DCSS users and LCSA Users with a CCSAS CSE-supplied desktop through a single point of contact. CCSAS customers, who include custodial and non-custodial parents, must access the Customer Service Center for questions or assistance. All help desk users will access Help Desk Support through the Level 1 services. This common process will maintain the proper call documentation and enable Level 1 Help Desk Representatives to resolve common problems using scripts. The Help Desk will be available to the CCSAS project staff and alliance staff 90 days after contract award to support CCSAS project planning and operations. CCSAS project staff, including application design, development and testing staff, will call the Help Desk for support of their desktop, standard desktop software, and the eight CSE environments. The Help Desk will not be used as a test recording or reporting facility during the application testing process. Those services will be coordinated by the alliance Test Manager.

Software User Manual

The alliance will develop a software user manual, which will contain system information and procedures to assist child support personnel in the performance of their job duties. This manual will be on-line. It will contain a search engine, glossary, index, and a substantial use of hyperlinks to maximize its usefulness. This manual will be organized

into logical modules based on system functionality. Development of this manual will largely be driven by the development of the system functionality and by the availability of the corresponding design documentation.

Communication Strategy with Development Team

In order to develop and deliver the best Software User Manual possible, strong lines of communication must be developed between the alliance's Training Team and Development Team. Because the Software User Manual needs to be based on the most up-to-the-minute design documentation, the Development Team will make this documentation available as soon as possible to the Software Users Manual and Training materials developers. Based on past experience, the reliance on manual dissemination of design documentation (either through hard copies or email attachments) is impractical, and inevitably major communication problems arise. Therefore, we suggest the "working" design documents be made available to the appropriate Software Users Manual/Training developers through a shared drive on a secure project server (or alternatively, on a secure web site.) To address security concerns, these documents can be made read-only for all but the designated personnel. We also suggest the use of automated email notification whenever new or revised documentation becomes available on the shared drive/web site to ensure that the Software Users Manual/Training developers receive notice in a timely manner.

Select appropriate development application/tool

Selecting the best organizational and conceptual approach to the Software User Manual should supercede the selection of the development application/tool at this time. Choosing a developmental tool for the Software User Manual is difficult until more is known about what kind of system interface is being developed. However based on our current knowledge, we recommend that the Software User Manual be HTML or Windows-based; then it can be an equally effective resource whether it is used as a stand-alone product or it is made accessible to users through hyperlinks in the context-sensitive Help messages.

Software User Manual Design/Develop/Test

Designing and organizing the Software User Manual will be one of the most crucial phases in its construction. The manual should be designed with a logical, modular organizational structure that is "open-ended" to allow for future additions and revisions. It should be divided into topics and subtopics, based on system functionality and an understanding of the CSE application workflow. By organizing the manual in this fashion, it will be equally useful to workers whether they have specialized or "cradle to grave" job tasks. After an organizational structure is created, authors can begin developing the content. This content will be based on system design documentation and on the workflow information that is determined from user focus groups and business process re-engineering. Because the Software User Manual will be a useful tool during and after the training process, its authors should work closely with the Training developers so that they produce a manual that consistently meshes with the classroom/e-learning materials. While the content is being developed, authors will also develop additional components, such as a glossary, index, and hyperlinks to make the manual a powerful,

integrated tool for the users. During and after the developmental phase, a quality assurance team will perform thorough testing to ensure the best product possible.

During the development of the Software User Manual, strong lines of communication must be maintained between the alliance team that is developing the Software Users Manual and development. Design documents should be made available through a shared drive on a secure server or a secure web site. A thorough review of the new and standardized CCSAS procedures is vital to the development of the Software Users Manual. More information about the application is needed before a development application/tool can be determined, but if possible, it should be integrated with context-sensitive Help. The organization and content of the Software User Manual should be determined mainly by the design documentation, user input, and work process re-engineering.

3.C.14.3.2 Approach to Developing and Meeting Help Desk Service Level Objectives

By using a flexible approach to Help Desk Services, the alliance can meet Help Desk Service Level Objectives by leveraging additional resources when they are needed.

Service Level Management is an alliance SMC Discipline. The CCSAS Executive Project Director and the alliance Project Executive have jointly established the specific Help Desk objectives. These Service Level Objectives may be modified later by agreement to include information assembled from a baseline of Help Desk performance in Production. Those Service Level Objectives will be documented in the Service Delivery Management Plan (CDL TM 076).

The alliance will use an automated Help Desk tool set for monitoring and meeting Help Desk Service Level Objectives. The Help Desk tool set includes problem reporting, Help Desk Service Level reporting, and a means for tracking problems through a closed-loop customer support process. Help Desk Service Level achievement will be documented in the Service Level Attainment Report.

As discussed earlier in this Section, the automated tracking system the alliance will use to log and monitor calls will provide regular reports for the CCSAS project staff – tracking both the numbers and types of calls and the progress toward meeting Service Level Objectives. The alliance will present Help Desk Service Level Attainment information to the CCSAS project staff on a monthly basis; we will maintain an ongoing discussion concerning those objectives.

The alliance has many methods for meeting Service Level Objectives. The Help Desk Leader will be responsible for meeting Help Desk Service Level Objectives and to develop plans to adjust service or processes in the event of consistently missed Service Level Objectives. Through Service Level Reporting the Help Desk Leader will maintain an understanding of the challenges of delivering CCSAS CSE Help Desk services. As part of the IBM Service Delivery Center the Help Desk Leader will be able to adjust resource allocations, add training for Level 1, 2 or 3 support, VRU parameters and settings, or Help Desk processes to better meet Service Level Objectives.

3.C.14.3.3 Approach to Training Help Desk Staff

The alliance has an ongoing commitment to advancing the specialized skills and expertise of our professionals. Our approach provides each IT professional with an outstanding opportunity for enhanced skill development that will qualify them for various roles in the delivery of quality services.

To continue our leadership role in the IT industry and provide employees that are qualified and proficient, the alliance has implemented a skills program that directly ties employee skills development and career growth to customer requirements and strategic business needs. Our skills framework allows employees to enhance their industry-oriented technical skills, based on the right mix of education, training, and experience.

To provide the CCSAS end user with knowledgeable Customer Support Representatives, each CSR will receive special training in one of the service areas provided under the contract. Initial training will involve training in Help Desk procedures - e.g., in the use of support tools and the knowledge base to support problem resolution. The Help Desk Training Coordinator will be responsible for identifying the training needs of Customer Support Representatives. The alliance Training Coordinator will work with other alliance members to identify available courses to provide that training or to develop new training specifically designed for Help Desk personnel.

3.C.14.3.4 Transitioning Help Desk Responsibilities to the State

The alliance utilizes the same transition planning approach to transition workload, or Help Desk Services, in or out of our facilities. The alliance would work with the CCSAS project to develop a Help Desk Transition Plan - including the identification of the transition time frame, the network configuration required for the transition period, the definition of transition roles and responsibilities, and the turnover of CCSAS-specific processes.

The alliance has determined that the CCSAS CSE Help Desk will be located at the Boulder Colorado IBM Service Delivery Center for the initial CCSAS CSE phases. This approach will provide the hours of service in a more cost effective manner than a new and dedicated facility. The CCSAS project staff will have escorted access to the CCSAS CSE Help Desk at all times for education and monitoring purposes.

Prior to transition back to the State the alliance will move the Help Desk to the alliance Project Management Office to facilitate training activities prior to the CCSAS project staff's takeover of Help Desk Services.

The Help Desk Transition will provide the following information and/or services to the Successor:

- Access to, and use of, CCSAS CSE Help Desk hardware, software, personnel, third parties, and other resources being provided to the CCSAS project as part of the transitioned CCSAS CSE Help Desk
- Information regarding the Help Desk services in order for the Successor to assume responsibility for, and continue the performance of, the Help Desk services in an orderly manner, so as to minimize operational disruptions
- Copies of documentation pertinent to CCSAS' operation used by the alliance. That documentation will include Help Desk Reports, documented Help Desk Procedures, Help Desk Open Problem Logs and Help Desk scripts
- The alliance will provide training for the automated call directing system and problem tracking system to the CCSAS project staff at the alliance Help Desk site during the transition of services from the alliance to the State (TM 14.3.6)
- Technical specifications, database definitions, and existing user documentation
- Reasonable access to the alliance personnel who were performing the Help Desk services

Upon transition back to the State, or to another Service Provider, the alliance will provide transition information and services in an appropriate and professional manner.

3C.14.3.5 Conclusion

The alliance provides Help Desk services to a large and diverse set of customers around the globe. The alliance has been providing IT Help Desk services internally for over 20 years. Since 1990, we have provided Help Desk support to commercial and governmental customers. Our customer base is drawn from both the public and private sector and includes a combination of large and very large installed bases. Garnering all our cumulative experience and expertise, the alliance will work to provide the CCSAS project with a readily adaptable, single point-of-contact Help Desk and an enterprise-wide problem management system for CCSAS CSE.

PART 2

Technical Management

Statements of Work & State Responsibilities

Statement of Work	State Responsibilities
TM 0 General Technical Management	
TM 0.1 The Business Partner shall conduct the Technical Management program in accordance with the Technical Management Approach. The Technical Management program shall meet the intent of IEEE 12207.	
TM 0.2 The Business Partner shall develop, implement, and monitor the processes and procedures necessary to implement the Technical Management plans.	
TM 0.3 The Business Partner shall conduct a System Engineering program in accordance with the Business Partner's System Engineering Management Plan (SEMP). (CDL TM 001) The System Engineering program shall meet the intent of IEEE 1220.	
TM 0.4 The Business Partner shall conduct a Configuration Management program that meets the intent of IEEE 1042.	
TM 0.5 The Business Partner shall conduct a Software Development program in accordance with the Business Partner's Software Development Plan (SDP). (CDL TM 002)	
TM 0.6 The Business Partner shall provide assistance to support the CCSAS project staff in the ACF certification process. This assistance shall include supporting State staff and Federal reviewers in planning for and conducting certification review(s) and Financial Test Deck testing; and assisting CCSAS staff in developing and implementing strategies to resolve certification issues.	The State will: - Lead and manage the activities associated with achieving ACF certification - Resolve State policies and/or statute issues that could create issues or ambiguities pertaining to federal requirements; - Proactively work with the federal government to secure waivers or clarifications necessary to achieve certification; - Request federal certification review; - Schedule and organize certification

Statement of Work	State Responsibilities
	planning and review meetings; - Review and finalize responses to federal questions resulting from the certification review, with appropriate consultation from the Business Partner; - Lead federal certification review sessions with the Business Partner attending and supporting the review; and - Coordinate and manage activities with the SDU vendor to verify that the SDU is operating sufficiently to reach ACF certification.
TM 0.7 The Business Partner shall develop and implement methods, processes, and procedures to verify that technical documentation matches the 'as built' CSE system. The Business Partner shall update and deliver any documentation found to not match the 'as built' CSE system.	For the purpose of developing interfaces, the State will provide the Business Partner with the currently available detailed design documentation from ARS and CASES.
TM 0.8 The Business Partner shall perform routine maintenance activities and cycle processing to support the operation of CSE environments.	
TM 0.9 The Business Partner shall perform requirements analysis and design for ARS and CASES system modifications for Version 1. The results shall be documented in the ARS Functional Requirements Design Document (CDL TM 101), and the CASES Functional Requirements Design Document. (CDL TM 102) The Business Partner shall participate in ARS and CASES testing, to include reviewing test documentation, witnessing tests, and conducting acceptance tests. The Business Partner shall advise the State on issues or problems encountered during ARS and CASES testing.	The State will conduct a consistency analysis review of the CASES and ARS systems. The State will provide the information necessary for the Business Partner to define the system requirements for revisions to ARS and CASES. The State will provide business and technical staff knowledgeable in the ARS and CASES systems to participate in requirements analysis and design. The State is responsible for the development and implementation of ARS and CASES system modifications and training materials, and will conduct training on the system modifications. The State will coordinate ARS and CASES testing activities. The State will participate in the acceptance tests conducted by the Business Partner for modifications to ARS and CASES.

Statement of Work	State Responsibilities
TM 0.10 The Business Partner shall support the State in the development of a single guideline calculator for Version 1.	The State is responsible for the procurement and implementation of a single guideline calculator and training on the use of the guideline calculator program.
TM 0.11 The Business Partner shall design, develop and implement a method for end-users to access and update POP data for Version 2 (updated per CR-00124a).	
TM 1 Introduction (Reserved)	
TM 2 Referenced Documents (Reserved)	
TM 3 Overview of Technical Management Approach (Reserved)	
TM 4 System Configuration Management	
TM 4.1 The Business Partner shall have change control procedures and processes in place that *…verify and validate changes to master files and application software (ACF H-3a).*	
TM 4.2 The Business Partner shall manage the system configuration in accordance with the System Configuration Management Plan. (CDL TM 003)	
TM 4.3 The Business Partner shall identify the items to be placed under configuration control and shall define when items are placed under configuration control. The Business Partner shall assign a project-unique identifier to each configuration item. The identification scheme shall be at the level at which entities will actually be controlled, for example, computer files, electronic media, documents, software units, configuration items. The identification scheme shall include the version/ revision/ release status of each entity.	
TM 4.4 The Business Partner shall use tools and processes to achieve an integrated management approach to capture, control and maintain the technical documentation, technical data, and technical manuals.	
TM 4.5 The Business Partner's System Configuration Management Procedures (CDL TM 004) shall *…ensure that only authorized changes are made to application software and that these changes are fully tested, approved, and migrated into production in a controlled manner (ACF H-3b).*	

Statement of Work	State Responsibilities
TM 4.6 The Business Partner shall prepare and maintain records of the configuration status of configuration items to provide …*an audit trail of all system maintenance (ACF H-3b).* These records shall be produced monthly, maintained for the life of the contract, and provided to the State project records center (updated per CR-00083-1).	
TM 4.7 The Business Partner shall use a version control tool that employs access control mechanisms. The Business Partner shall provide the State access to the Business Partner's version control tool.	
TM 4.8 Business Partner shall use a version of the Microsoft Office Suite that is compatible with the State's Microsoft Office Suite.	
TM 4.9 Deleted.	
TM 4.10 Deleted.	
TM 5 Requirement Management	
TM 5.1 Deleted.	
TM 5.2 The Business Partner shall develop and implement a requirement management program in accordance with the Requirements Management Plan (CDL TM 006). The Business Partner's requirement management program shall include processes to manage Business, System and Software Requirements, prior to and after establishment of the baseline.	The State will assist the Business Partner in implementing and administering the requirement management process.
TM 6 Technical Risk Management (Reserved)	
TM 7 Knowledge Management	
TM 7.1 The Business Partner shall conduct a Knowledge Management program in accordance with the Knowledge Management Plan. (CDL TM 007) Knowledge management focuses on technical knowledge and skills which only relate to the specific needs of the individual roles and responsibilities. Training in non-technical fields, for example leadership development, shall not be provided. The Business Partner shall not be responsible for knowledge transfer on CASES.	The State is responsible for developing and verifying that State employees participating in knowledge management activities have the appropriate skill set for the activity and meet course requirements. The State will participate in the development, review, and assessment survey of Individual Development Plans for State staff. The State resources will participate in the knowledge management activities including self-assessment, development of Individual Development Plans, assessment surveys, and feedback mechanisms.

Statement of Work	State Responsibilities
TM 7.2 The Business Partner shall provide the State CCSAS project staff with formal, informal, or on the job training for software development platforms and tools. The Business Partner and State shall develop a mutually agreed schedule for knowledge management training. The Business Partner shall also provide educational packages for State staff attending formal courses from IBM Learning Services. Courses shall be offered in California.	The State resources will make a good faith effort to attend the knowledge management training courses and/or sessions as scheduled. The State will be responsible for providing vendor training for additional or new staff in excess of the 47 allocated. The State will be responsible for any travel expenses related to attending formal vendor training offered by IBM Learning Services.
TM 7.3 The Business Partner shall provide formal, informal, and on the job training for software development platforms and tools. The courses shall be focused on the customized usage and implementation of CCSAS software and tools as it relates to project-specific methodologies, procedures and standards. The course information shall be documented in the Software Development Tools Training Materials. (CDL TM 008) The Software Development Tools Training Materials shall be available during the life of the project. The Business Partner's custom-developed training materials shall be maintained to reflect the current development approach and use of the software development tools. The formal classroom training referenced in this paragraph shall have materials that support repeatable training throughout the life of the project.	New State resources added to the project will work with the Business Partner's Knowledge Management Lead or personnel to develop an Individual Development Plan and review the existing training documentation for courses already completed.
TM 7.4 The Business Partner shall employ standard training techniques when formal classroom training is provided.	
TM 7.5 Deleted.	
TM 7.6 The Business Partner shall provide a method for the State to provide feedback on the effectiveness of the Knowledge Management program.	The State will provide feedback against the measurable criteria of the Knowledge Management Plan concerning the effectiveness of the Knowledge Management program.
TM 7.7 The Business Partner shall establish specific, measurable criteria for assessing the effectiveness of the Business Partner's Knowledge Management program, and shall modify the program to improve areas of weakness.	
TM 8 Technical Reviews	

Statement of Work	State Responsibilities
TM 8.1 The Business Partner shall plan and conduct joint technical reviews at locations and dates proposed by the Business Partner and approved by the State. The reviews shall be a joint effort by Business Partner and State representatives. The Business Partner and State shall co-chair the reviews and the Business Partner shall lead the effort to implement the decisions made as a result of the review.	
TM 8.2 Deleted.	
TM 8.3 Technical reviews shall be conducted to assess the degree of completion of technical efforts related to major milestones before proceeding with further dependent technical effort. The Business Partner shall prepare Technical Review Agenda (CDL TM 096) and shall disseminate the review materials and accepted agenda materials a minimum of 10 State business days prior to each review. The Business Partner shall prepare and disseminate Technical Review Minutes as set forth in the Systems Engineering Management Plan (SEMP). Specific reviews are listed in SOW items 8.3.1 through 8.3.11. These reviews shall be conducted when the entry criteria specified in the System Engineering Management Plan (SEMP) (CDL TM 001) are satisfied.	
TM 8.3.1 System Requirements Review (SRR). The objective of this review is to ascertain the adequacy of the Business Partner's efforts in capturing, specifying, and documenting system requirements.	
TM 8.3.2 Deleted.	
TM 8.3.3 System Design Review (SDR). This review shall be conducted to evaluate the optimization, correlation, completeness, and risks associated with the allocated requirements. The review evaluates a summary review of the system engineering process that produced the allocated technical requirements, and of the engineering planning for the next phase of effort.	
TM 8.3.4 Deleted.	
TM 8.3.5 Software Requirement Review (SWRR). The objective of this review is to ascertain the adequacy of the Business Partner's efforts in defining software requirements.	
TM 8.3.6 Deleted.	

Statement of Work	State Responsibilities
TM 8.3.7 Software Design Review (SWDR). This review shall be conducted for each configuration item or aggregate of configuration items to (1) evaluate the progress, technical adequacy, and risk resolution (on a technical, cost, and schedule basis) of the selected design approach, (2) determine software design compatibility with performance and business requirements, and (3) establish the existence and compatibility of the interfaces. The SWDR shall also include a review of the logical data model.	
TM 8.3.8 Deleted.	
TM 8.3.9-(SOW deleted per CR-C-01012)	
TM 8.3.10 Deleted.	
TM 8.3.11 Operational Readiness Assessment and Review (ORAR). This review is intended to determine the status of completion of the specific actions that must be satisfactorily accomplished prior to executing a production go-ahead decision. This review will assess the risk in exercising the go-ahead decision	
TM 8.3.12 Deleted.	
TM 8.4 The Business Partner shall include a high-level analysis of the business process re-engineering impact as it relates to the scope of the review.	
TM 9 Technical Quality Management	
TM 9.1 The Business Partner shall conduct a Software Quality program in accordance with the Software Quality Plan. (CDL TM 009)	
TM 10 Problem Resolution	
TM 10.1 The Business Partner shall plan and conduct problem management activities in accordance with the Problem Resolution Management Plan. (CDL TM 010)	
TM 10.2 The Business Partner shall track, manage, and report system development, system implementation, and production support problems and their resolution. The Business Partner shall make the problem resolution status reports available to the State.	The State will track, manage, and report system development, system implementation, and production support problems and their resolution for problems associated with the ARS and CASES platforms.

Statement of Work	State Responsibilities
TM 10.3 The selected problem management software tool shall provide the ability to produce pre-defined and ad-hoc statistical reports. The problem management software tool shall employ access control mechanisms. The Business Partner shall provide the State access to the problem management software tool.	
TM 11 System Life Cycle Model (Reserved)	
TM 12 System Development	
TM 12.0 General System Development	
TM 12.0.1 - The Business Partner's Business Solution shall be used as the basis for System Development. The Business Partner shall develop a CSE system that is compliant with the Business Requirements List (BRL). The Business Partner shall document traceability between the Business Partner's solution and the Business Requirements. The Business Partner shall perform traceability between levels of requirements, between requirements and design, between design and software that implements the design, and between requirements and test cases. The Business Partner's requirements management tool shall employ access control mechanisms. The Business Partner shall provide the State access to the Business Partner's requirements management tool.	The State will assist the Business Partner with configuration of the requirements management tool to facilitate integration with DOORS.
TM 12.0.2 The Base Business Concepts shall be incorporated within the Business Partner's solution design as the basis for the information architecture.	
TM 12.0.3 The Business Partner shall develop and implement the user interface in accordance with the User Interface Standard. (CDL TM 013)	State and LCSA will identify users to participate in Low Fidelity Testing.
TM 12.0.4 The Business Partner shall prepare a Software Product Specification (SPS) and shall generate the executable software in accordance with the SPS. (CDL TM 014)	
TM 12.0.5 The Business Partner shall use a project modeling tool(s), employing UML, to capture, record, verify and version business processes and business rules, and shall integrate them into the appropriate specification and design documents. The project modeling tool(s) shall employ access control mechanisms. The Business Partner shall	

Statement of Work	State Responsibilities
provide the State access to the Business Partner's project modeling tool(s).	
TM 12.0.6 Deleted.	
TM 12.0.7 The Business Partner shall participate in State-managed forms re-engineering activities, to include participating in work groups to determine forms needs and achieve user buy-in.	The State will direct form updates and changes. This will include language, data, formatting, and layout. The State will develop the draft forms in accordance with State-defined forms standards.
TM 12.0.8 The Business Partner shall use the State-developed draft forms as the layout for forms design and development. The Business Partner shall perform mapping from the data elements on the form to data elements in the database.	
TM 12.0.9 The Business Partner shall use tool features as mutually agreed to by the State and the Business Partner based on joint professional experience and the suitability for the project's needs.	
TM 12.1 System Requirements Analysis	
TM 12.1.1 The Business Partner shall conduct working groups with representatives from the user community to gain an understanding of user needs.	
TM 12.1.2 The Business Partner shall perform system requirements analysis and document the results in the System / Subsystem Specification (SSS). (CDL TM 016-1, TM 016-2)	
TM 12.1.3 The Business Partner shall conduct a training session for requirements analysis roles, practices and procedures in accordance with the TMA. This training may be conducted prior to the commencement of a formal Knowledge Management program.	

TM 12.2 System Architecture Design	
TM 12.2.1 The Business Partner shall define and record the architectural design of the system in the System / Subsystem Design Description (SSDD). (CDL TM 017-1, TM 017-2)	The State will make a good faith effort to provide the same experienced CCSAS project staff and LCSA personnel to participate in System Requirements through Software Design.
TM 12.2.2 The Business Partner shall define and record the detailed infrastructure design in the Technical Infrastructure Description (TID). (CDL TM 018) The Business Partner shall incorporate the State-provided WAN detailed infrastructure design into the TID.	The State will create the WAN detailed infrastructure design.
TM 12.2.3 The Business Partner shall allocate performance requirements and monitor system performance.	
TM 12.2.4 The Business Partner shall build and test the CSE system environments, including a CSE Practice Environment as more particulary described in TM 064, User Training Plan (updated per CR-2-01124).	
TM 12.2.5 The Business Partner shall participate and assist in the design of the WAN and incorporate the State-provided WAN design into the System/Subsystem Design Description. (SSDD) (CDL TM 017-1, TM 017-2)	The State will implement and manage a WAN to support the following CCSAS CSE Project, LCSA, State, and Child Support Program sites: ▪ LCSA offices ▪ Auxiliary county sites that use CSE applications ▪ IBM e-business Hosting Center in San Jose ▪ Regional training centers ▪ Customer Service Support Center(s) ▪ DCSS ▪ CCSAS CSE Project site(s) ▪ CCR ▪ IBM Help Desk in Boulder, Colorado via the HHSDC network The State will be provide, install and configure the circuits and routers necessary to connect the above sites to the HHSDC network. The State, working in conjunction with the Business Partner, will design the WAN to meet the performance requirements agreed to between the State and the

	Business Partner as defined in the Performance and Capacity Management Plan (CDL TM 86-1, TM 86-2). The State will proactively monitor the WAN, circuits and routers; provide Level 2 Help Desk support for the network; and provide proactive notification to the CSE Help Desk and LCSA Help Desk when network problems are identified. The State will provide for the administration, maintenance and repair of circuits and routers installed to connect the above sites.
TM 12.3 Software Requirements Analysis	
TM 12.3.1 The Business Partner shall define and document the software requirements in the Software Requirements Specification (SRS). (CDL TM 020-1, TM 020-2)	
TM 12.4 Software Architecture Design	
TM 12.4.1 The Business Partner shall define and record the software architectural design in the Software Design Description (SDD). (CDL TM 021-1, TM 021-2)	
TM 12.5 Software Detailed Design	
TM 12.5.1 The Business Partner shall define and record the software detailed design. The Business Partner shall make the results of detailed design activities available to the State. The Business Partner shall version control detailed software design work products prior to establishing the formal baseline. The Business Partner shall establish the configuration managed baseline of detailed software design work products at State acceptance of the System Verification Test Report (CDL TM 035) (updated per CR-00124a).	
TM 12.5.2 The Business Partner shall conduct periodic software development reviews in accordance with the Software Development Plan. (CDL TM 002)	
TM 12.6 Database Design and Development	
TM 12.6.1 The Business Partner shall produce and execute a Database Development Plan. (CDL TM 023)	

TM 12.6.2 The Business Partner shall design and record database access mechanisms in the Software Design Description. (CDL TM 021-1, TM 021-2)	
TM 12.6.3 The Business Partner shall produce and execute the System Administration Plan. (CDL TM 025)	
TM 12.6.4 Deleted.	
TM 12.6.5 The Business Partner shall configuration manage scripts, including operating system specific environment configuration / alteration script(s), data definition language creation script(s), database configuration alteration script(s), in accordance with the System Configuration Management Plan (CDL TM 003) and the change control process.	
TM 12.6.6 The Business Partner shall use readily available commercial software modeling tool(s) that support software and database development.	
TM 12.6.7 The Business Partner shall use a modeling tool(s) that provides the following features:	
TM 12.6.7.1 A standardized, publicly available modeling language.	
TM 12.6.7.2 Definition of relationships.	
TM 12.6.7.3 Support for integration between requirements, development, testing, and version control.	
TM 12.6.7.4 Support for selected programming languages.	
TM 12.6.7.5 Support for both forward and reverse engineering.	
TM 12.6.7.6 Storage of information on database access and modification characteristics including estimated frequency of the characteristic's use.	
TM 12.6.7.7 Interfaces to more than one readily available robust DBMS for repository purposes.	
TM 12.6.8 The Business Partner shall use the database modeling tool(s) to produce data definition language scripts.	
TM 12.6.9 The Business Partner shall make changes to the database through the database modeling tool, and shall keep all database scripts current.	

TM 12.6.10 The Business Partner shall use a modeling tool(s) and procedures to analyze the impact of changes.	
TM 12.6.11 Throughout the life of the project life cycle, the Business Partner shall develop and maintain data models with a minimum of two different perspectives into the system:	
TM 12.6.11.1 A logical data model. The Business Partner shall make the logical data model available to the State. The Business Partner shall establish the configuration managed baseline of the logical data model at acceptance of the Software Design Description. (CDL TM 021-1, TM 021-2)	
TM.12.6.11.2 A physical data model containing information to describe and create / recreate actual physical database schema. The Business Partner shall make the physical data model available to the State. The Business Partner shall establish the configuration managed baseline of the physical data model at State acceptance of the System Verification Test Report (CDL TM 035) (updated per CR-00124a).	
TM 12.6.12 The Business Partner shall use a modeling tool(s) that supports capture, record, verify synchronization, and versioning of database and system metadata.	
TM 12.6.13 The Business Partner shall develop or use a standard model of metadata management as documented in the Software Development Plan (SDP). (CDL TM 002)	
TM 12.6.14 The Business Partner shall develop or use a standard model of metadata interchange as documented in the Software Development Plan (SDP). (CDL TM 002)	
TM 12.7 Software Coding	
TM 12.7.1 The Business Partner shall develop software code in accordance with the Business Partner's software coding standards and the appropriate requirements specifications.	
TM 12.7.2 The Business Partner shall use debugging technologies that are capable of detecting, diagnosing, and reporting multiple classes of errors.	

TM 12.8 System Integration	
TM 12.8.1 The CSE Business Partner shall conduct system integration activities for the CCSAS CSE system, System integration responsibilities include identifying CSE data needs such that the CCSAS CSE is the system of record for child support data, conducting interface requirement definition, design, test planning, and test evaluation meetings and coordinating the analysis and resolution of problems involving external CSE system interfaces.	The State will be the business and technical point of contact (liaison) for the Business Partner for external interfaces (e.g., ACF, LCSA, DMV, etc.). The State will facilitate the resolution of issues with external entities' participation in system integration activities (e.g. requirements definition, design, testing.)
TM 12.8.2 The Business Partner shall design, integrate, test, and operate the CSE portion of the CSE / SDU interfaces. The Business Partner shall establish a point of contact for external interface agencies to report interface problems or operational impacts that may affect data exchange timeframes.	The State will coordinate activities with external entities required to create and implement the external system interfaces and facilitate the resolution of issues.
TM 12.8.3 The Business Partner shall, in coordination with the SDU vendor, plan and conduct the CSE / SDU integration activities in accordance with the SDU Integration Plan. (CDL TM 029)	
TM 12.8.4 The Business Partner shall participate in SDU implementation activities as they relate to processes affecting both systems to include attending SDU planning meetings, technical reviews, and supporting testing activities. The Business Partner shall review SDU developed interface definition document(s) for impacts to CSE data needs and the CSE / SDU interface.	The State will resolve issues with the SDU Business Partner participation in integration activities.
TM 12.8.5 The Business Partner shall define the CSE / SDU interface such that the CSE is the system of record for child support data; business rules and CSE / SDU business processes affecting both systems are integrated; and CSE data is available to the SDU to support SDU business processes.	
TM 12.8.6 The Business Partner shall define the required data exchange specifications for the CSE/SDU interface to include allocating the two State business days for disbursement between the CSE and SDU. The Business Partner shall document the interface requirements and design in the CSE / SDU Interface Design Description. (CDL TM 030-1, TM 030-2)	The State will validate that the SDU Business Partner builds and tests the SDU portion of the CSE / SDU interface in accordance with the requirements documented in the CSE / SDU Interface Design Description. (CDL TM 030-1, TM 030-2)
TM 12.8.7 Deleted.	

TM 12.8.8 The Business Partner shall conduct CSE / SDU transmission audits for the CSE portion of the CSE/SDU interface. These transmission audits, which may be automated, consist of record counts, reconciliation of payment data, and status of file processing activities. The Business Partner shall make the transmission audits and reconciliations available to the State upon request.	
TM 12.8.9 The Business Partner shall, when establishing new interfaces, leverage existing data interchange interfaces where appropriate and use standardized interfaces where appropriate. The Business Partner shall strive to extend interchange reach using evolving standards of universal format for structured documents and data, to the extent that these standards: are stable, are supported by CCSAS project tools, and fit within the proposed CCSAS CSE architecture.	If problems with cooperation from the interfacing entity occur, the State will coordinate to enable joint development and testing activities to occur.
TM 12.8.10 The Business Partner shall review the Interagency Agreements and External Agency Interface Agreements for consistency with the Business Partner's requirements and design and shall communicate issues to the State.	The State will negotiate and maintain interagency and External Agency Interface agreements. The State will provide Interagency Agreements and External Agency Interface Agreements to the Business Partner for review. If the agreements do not contain sufficient detail, the State will provide this detail, to the extent possible. The State will be responsible for resolving identified inconsistencies in the agreements.
TM 12.8.11 The Business Partner shall develop the CSE portion of interfaces with external entities in accordance with the Interagency Agreements and External Agency Interface Agreements.	The State will coordinate external agency participation in technical meetings and reviews. The State will coordinate the involvement and scheduling of subject matter experts during the external entity interface design sessions.
TM 12.8.12 The Business Partner shall define the required data exchange specifications for CSE external interfaces and produce an External Entity Interface Design Description (EE IDD) (CDL TM 031-1, TM 031-2). The Business Partner may reference federal interface guidance documents for the Federal Case Registry, Financial Management Systems and CSENet interfaces in lieu of incorporating these documents into the EE IDD.	

TM 12.8.13 The Business Partner shall participate in external entity interface definition meetings as requested by the State. Participation shall include the preparation of technical materials, the identification of issues, risks and schedule impacts, and resolving and reporting on assigned action items.	
TM 12.9 Test Management	
TM 12.9.1 The Business Partner shall conduct a testing program in accordance with the Master Test Plan. (CDL TM 032)	
TM 12.9.2 The Business Partner shall plan and conduct testing in accordance with the Software Test Plan (STP). (CDL TM 033)	
TM 12.9.3 The Business Partner shall plan, execute and evaluate the following types of tests:	
TM 12.9.3.1 Unit Tests.	
TM 12.9.3.2 Unit Integration Tests.	
TM 12.9.3.3 System Tests. System testing shall include functional, integration, performance and interface testing.	The State will coordinate the participation of external entities in the planning, execution and evaluation of system tests. The State will coordinate with external entities to obtain test data necessary for system testing.
TM 12.9.3.4 System Verification Tests. System verification testing shall include functional, integration, and interface testing. Interface testing shall be conducted using interfacing entity production data when available.	The State will coordinate the participation of external entities in the planning, execution and evaluation of System Verification tests. The State will coordinate with external entities to obtain test data necessary for System Verification testing.
TM 12.9.3.5 System Performance tests.	
TM 12.9.4 The Business Partner shall exclude personnel who are involved in the development of the software from the creation of test scripts for system verification testing. The Business Partner shall exclude the originating software developer(s) from the dry run and the formal witnessed execution and evaluation of the test scripts.	

TM 12.9.5 The Business Partner shall define and record test preparations, test cases and test procedures to be used during System Test and the traceability between test cases and the software requirements. The Business Partner shall analyze and record the results of system testing. The Business Partner shall version control testing work products and makes the testing work products available to the State.	
TM 12.9.6 The Business Partner shall conduct the formal System Verification Test in accordance with the System Verification Test Description (CDL TM 034-1, TM 034-2). Prior to the formal System Verification Test, the Business Partner shall conduct a dry run of the System Verification Test Description (CDL TM 034-2) procedures at the CCSAS project site to verify that they are complete and accurate and that the system is ready for witnessed testing. The Business Partner shall update the software test cases and procedures as appropriate.	State staff with knowledge of the requirements being tested will participate in witnessed system verification testing.
TM 12.9.6.V1 The Business Partner shall conduct System Verification Test in accordance with the Software Test Plan (CDL TM 033) (added per CR-00124a).	State staff with knowledge of the requirements being tested will participate in system verification testing. State staff will work jointly with Business Partner during the execution of the System Verification Test. The State will coordinate the participation of external entities in the planning, execution, and evaluation of system test. The State will coordinate with external entities to obtain all test data necessary for system testing.
TM 12.9.7 The Business Partner shall document the results of formal System Verification testing in a System Verification Test Report. (CDL TM 035)	
TM 12.9.7.V1 The Business Partner shall report the results of the system verification test in the System Verification Test Report (CDL TM 035) (added per CR-00124a)	
TM 12.9.8 The Business Partner shall use automated tools for testing as specified in the Software Test Plan. (CDL TM 033)	

TM 12.9.9 The Business Partner shall use converted production data as test data when appropriate. The production data shall be run through conversion programs before testing occurs as specified in the Software Test Plan. (CDL TM 033)	The State will provide the Business Partner with unconditioned source data to be used for testing.
TM 12.9.10 The Business Partner shall conduct *…all testing of programs using test data as opposed to 'live (production) data' (ACF H-3d).*	The State will provide the Business Partner with unconditioned source data to be used for testing.
TM 12.9.11 The Business Partner shall include regression testing during system testing and System Verification testing.	
TM 12.9.12 The Business Partner shall conduct the formal System Performance test in accordance with the System Performance Test Description. (CDL TM 099-1, TM 099-2) The Business Partner shall make performance test results available to the State.	
TM 12.9.13 The Business Partner shall provide technical support for the State during the planning, conducting and evaluation of System Qualification Testing (i.e. acceptance testing).	The State may, at the State's discretion, conduct System Qualification Testing. System Qualification Testing, if conducted, will be in accordance with the State approved CCSAS project schedule. Any System Qualification Testing and associated activities will take place at the CCSAS project site in Sacramento.
TM 12.9.13.V1 The Business Partner will provide support equivalent to not less than four (4) staff for two months to support the State in the System Qualification Testing (SQT). This support will include assisting the State in the development of the SQT Plan, in the execution of the SQT, and in the evaluation of the test results (added per CR-00124a).	The System Qualification Testing (SQT) will be planned and conducted by the State.
TM 12.9.14 The Business Partner shall develop and implement procedures for unit and unit integration testing. The Business Partner shall document the results of Unit Tests and Unit Integration Tests, which shall be available for State review. The Business Partner shall tailor the intent of IEEE 12207.2 1997 5.3.7 & 5.3.8 to be consistent with the TMA and shall document the tailored approach in the Software Development Plan (SDP). (CDL TM 002)	
TM 12.9.15 Deleted.	
TM 12.9.16 Deleted.	

TM 13 System Implementation	
TM 13.0 General System Implementation	
TM 13.0.1 The Business Partner shall assess the pre-implementation readiness of each LCSA and DCSS and shall document the status in a Pre-Implementation Readiness Assessment. (CDL TM 038) The Business Partner shall conduct an implementation readiness review ten days prior to cutover at each LCSA and DCSS.	The State and LCSAs will provide timely access to the site, systems, materials, and appropriate resources needed for the assessment. The State will participate in the implementation readiness review and issue an implementation go / no go decision.
TM 13.0.2 The Business Partner shall conduct a post-implementation review following implementation at each LCSA and DCSS, and shall prepare a Post-Implementation Review Report. (CDL TM 039)	
TM 13.1 Conversion Management	
TM 13.1.1 The Business Partner shall convert summary case balances and data for open child support cases stored in the KIDZ andSTAR/KIDS consortia systems necessary to transition the Phase I LCSAs to CASES using a common set of conversion plans, data maps, and derivation programs for each consortium. The Business Partner shall convert Non-IV-D data from the NICE database to the CCSAS CSE Statewide Services database. The Business Partner shall convert summary case balances and data for open child support cases stored in the CASES and ARS consortia systems necessary to transition the Phase II LCSAs to the new system using a common set of conversion plans, data maps, and derivation programs for each consortium. The Business Partner shall convert 'as is' data from the POP, IDB, and CCR databases to the CCSAS CSE. The Business Partner shall load the converted data into the target database environments in support of the testing, validation, and production cutover activities. The Business Partner shall perform dry run testing against converted data by developing and executing test scenarios.	The State will be responsible for providing data conversion hardware/software environments for each Phase I LCSA. The State will provide extracts of their data in accordance with the data maps. Conversion data cleansing and post conversion data cleanup activities for the consortia, NICE, CCR, IDB and POP databases will be the responsibility of the State. The State will be responsible for all manual data entry of information required for conversion but not available from automated sources. The State will validate conversion results and perform manual data entry of cases that did not convert. The State will coordinate activities, respond to issues and resolve problems regarding the transition to CASES and non-IV-D conversion for Phase I, and CSE, CCR and IDB, and POP for Phase II. The State will support conversion activities such as data mapping, data cleansing, data acceptance testing and cutover.

TM 13.1.2 The Business Partner shall plan, conduct, and execute legacy data archive activities in accordance with the Legacy Data Archive Plan. (CDL TM 040) Legacy data archive implementation activities shall include the design and build of the archive database in accordance with the Legacy Database Archive Design Description. (CDL TM 100); the design and build of application modules for access to the archive database functions; mapping and transfer of archive data; the design and build of archive automated conversion tools; and the development of standard format descriptions. The Business Partner shall provide one FTE to assist the State by performing tasks such as responding to data mapping issues and interpreting the results of errors generated by the archive ACS. The Business Partner shall provide historical data conversion support beginning in month six of the project and continuing through month sixty. The Business Partner shall develop and deliver training materials for the use of the archive database and transfer and use of archived data. The Business Partner shall prepare and execute archive extract programs for the ARS, CASES, KIDZ and STAR/KIDS legacy systems and shall create the load file for the legacy data archive. The Business Partner shall deliver the populated and operational legacy data archive no later than one year following full implementation of the CSE System Version 2. The Business Partner shall design and develop the Legacy Data Archive in accordance with the Legacy Database Archive Design Description. (CDL TM 100) The Business Partner shall provide to the State the support necessary to ensure that the Legacy Data Archive accepts data from a county which was not coverted by the Business Partner, agreed to by the Parties.	The State will support legacy data activities such as data mapping, data cleansing, data acceptance testing and cutover for ARS, CASES, KIDZ and STAR/KIDS. The State and Business Partner will develop mutually agreed to business rules and data conversion requirements for the legacy data archive. The State will prepare legacy data archive extract files for the State-selected child support program legacy systems other than ARS, CASES, KIDZ and STAR/KIDS in accordance with the format provided by the Business Partner.
TM 13.1.3 The Business Partner shall define conversion requirements and produce a Data Conversion Requirements Document. (CDL TM 041-1, TM 041-2)	

TM 13.1.4. The Business Partner shall plan and conduct conversion activities in accordance with the Data Conversion Requirements Document (CDL TM 041-1, TM 041-2) and the Data Conversion Plan (CDL TM 042-1, TM 042-2). Conversion activities shall include development of conversion software tools, development of data conversion requirements, mapping of current system data to target system, identification of pre-cutover data clean-up and post-cutover data fallout, pre-cutover testing of converted data, conversion of image data, and archival of historical data.	
TM 13.1.5 The Business Partner shall develop and implement procedures to test converted data prior to cutover. The Business Partner shall develop test scripts and document test results. The test scripts and test results shall be available to the State upon request.	
TM 13.1.6 Deleted.	
TM 13.1.7 The Business Partner shall produce a data map for each converted system that maps current system data to the target system data. The finalized data map shall be provided as an attachment to the Post Implementation Review Report. (CDL TM 039)	
TM 13.1.8 Deleted.	
TM 13.1.9 The Business Partner shall develop and implement procedures to test the conversion software tool. The Business Partner shall develop test scripts and document test results in accordance with the Data Conversion Plan. (CDL TM 042-1, TM 042-2) The test scripts and test results shall be available to the State upon request.	
TM 13.1.10 Deleted.	
TM 13.1.11 The Business Partner shall analyze post-cutover data fallout and support State fallout resolution activities by identifying and prioritizing data that did not convert, in accordance with the Data Conversion Plan. (CDL TM 042-1, TM 042-2)	
TM 13.1.12 Deleted.	
TM 13.1.13 The Business Partner shall establish pre-conversion trial balances and post-conversion trial balances, and shall provide the results of the reconciliation process in accordance with the Data Conversion Plan. (CDL TM 042-1, TM 042-2)	The State will support the trial balance reconciliation effort.
TM 13.1.14 Deleted	

TM 13.1.15 Deleted.	
TM 13.1.16 The Business Partner shall incorporate lessons learned from each conversion into the planning and management of upcoming conversions.	
TM 13.1.17 Deleted.	
TM 13.1.18 The Business Partner shall maintain and operate the existing CASES Automated Conversion System (ACS).	The State will provide the existing VAX-based CASES ACS infrastructure, and additional disk storage as needed .
TM 13.1.19 The Business Partner shall prepare and execute archive derivation programs for the ARS, CASES, KIDZ, and STAR/KIDS legacy systems and shall create the load file for the legacy data archive.	The State will provide extracts of data bound for the legacy data archive from the ARS, CASES, KIDZ, and STAR/KIDS systems.
TM 13.2 Transition Management	
TM 13.2.1 The Business Partner shall develop an Outreach Plan (CDL TM 053) and shall coordinate and support Outreach program activities in accordance with the Outreach Plan. (CDL TM 053)	The State will provide education/outreach in accordance with the Outreach Plan (CDL TM 053). The State will provide the resources to review and approve outreach materials; deliver statewide outreach; and address stakeholder initiated program policy and procedural questions. The State will coordinate the development of CCSAS project outreach pages on the DCSS Website targeted to the general public. The State will customize the templates and materials to deliver local outreach information to external entities including customers. The State will take over outreach and communication activities following the full system implementation.
TM 13.2.2 The Business Partner shall develop and execute an Orientation Plan (CDL TM 054) and publish an Orientation Guide (CDL TM 055). The Business Partner shall facilitate and support the conduct of new-system orientation sessions prior to the system training for users. For Phase I and Phase II, the Business Partner shall support one orientation session for each LCSA with 100 users or less; two orientation sessions for LCSAs with 101 to 200 users; three orientation sessions for LCSAs with 201 to 400 users; four orientation sessions for LCSAs with 401 to 500 users; and six orientation sessions for	The State will make staff available who are knowledgeable in consortia application business processes to participate in the development of the baseline 'As Is' and 'To Be' business processes. The State will provide resources for development of the CCSAS Business Model who are familiar with the system requirements and design. The State will make decisions on how to integrate the CASES and CCSAS CSE functionality into site-specific operations and documentation. The State will incorporate the To-Be

LCSAs with more than 500 users. For Phase I, orientation activities are targeted to the staff of the 14 counties that transition from KIDZ, and STAR/KIDS to CASES. During Phase I, the Business Partner shall provide general orientation for all 58 counties and DCSS with regard to changes associated with the implementation of the Statewide Services component. The Business Partner shall update the baseline 'To Be' system functionality maps for CASES if any system changes are implemented after the Orientation Guide (CDL TM 055) is accepted.	business processes into LCSA's local operations and lead the conduct of a user orientation at each LCSA. The State will be responsible for updating the baseline CCSAS CSE business process documentation in the Orientation Guide for policy and procedural changes that are implemented after the Orientation Guide is accepted. The State will promptly notify the Business Partner of these changes. The State will be responsible for promptly providing decisions and executing actions with regard to the use of CASES and the integration of CASES functionality into their local operations, based on the system functionality maps provided by the Business Partner.
TM 13.2.3 The Business Partner shall create a Model Office as described in the Orientation Plan (CDL TM 054) and shall use it during the Version 2 orientation to demonstrate the interplay of the new statewide system software and any revised business processes. The model office shall be configured to support up to 20 concurrent users and shall provide the same functionality as that provided to an LCSA site.	
TM 13.2.4 The Business Partner shall notify the State's designated contact person of conversion problems that may impact Customer Service.	
TM 13.2.5 The Business Partner shall support the planning and activities associated with the transition of the KIDZ, and, STAR/KIDS consortia to the CASES system and the phasing in of functionality associated with the implementation of Statewide Services in accordance with the Business Process Transition Plan, (CDL TM 056) The Business Partner shall plan and conduct the activities associated with the transition of ARS and CASES consortia to the new CCSAS CSE System in accordance with the Business Process Transition Plan. (CDL TM 056)	The State will provide staff knowledgeable in IDB / CCR processes, organization, and data to support the transitions of these business functions and will be responsible for the transition of State personnel. The State will provide the resources necessary to complete the transition activities to implement CASES and the CCSAS CSE in accordance with the Business Process Transition Plan. (CDL TM 056)
TM 13.2.6 The Business Partner shall produce an IDB Transition Analysis Report. (CDL TM 057)	
TM 13.2.7 The Business Partner shall conduct IDB transition activities in accordance with the Business Process Transition Plan. (CDL TM 056)	

TM 13.2.8 The Business Partner shall produce a CCR Transition Analysis Report. (CDL TM 059)	
TM 13.2.9 The Business Partner shall conduct CCR transition activities in accordance with the Business Process Transition Plan. (CDL TM 056)	
TM 13.2.10 The Business Partner shall produce and execute an On-Site Implementation Support Plan. (CDL TM 061) The Business Partner shall provide staff at the proposed number of staff, ratios and duration for on-site support as defined in the TMA. The Business Partner staff for On-Site Support shall only be available during on-site business hours.	The State will provide staff at the proposed number of staff, ratios and duration for on-site support as defined in the TMA. Following the two-month period of on-site support provided by the Business Partner, the State becomes responsible for on-site support. The State will be responsible for identifying and gathering any site-specific data such as county parameters and user profile data and providing the data in aBbusiness Partner-defined format.
TM 13.2.11 The Business Partner shall convert the consortia member LCSAs prior to converting the lead consortia LCSA.	
TM 13.2.12 The Business Partner shall perform the activities necessary to identify and produce system functionality mapping documents for Phase I. The system functionality maps shall identify differences in system functionality that may exist between CASES and KIDZ and STAR/KIDS. The documents shall be incorporated into the Orientation Guide (CDL TM 055) for use by the Phase I LCSAs in planning the transition to CASES. For Phase II, the Business Partner shall participate in State child support business standardization activities by supporting the development and implementation of the CCSAS Business Model, which is documented in the Orientation Guide. (CDL TM 055)	The State and DCSS leadership will provide direction for the CCSAS Business Model and confirm the business processes. The State will provide resources to participate in the development of the CCSAS Business Model. The State will designate a representative to lead the process of developing the CCSAS Business Model who is knowledgeable about the State's desired uniform business processes.
TM 13.2.13 The Business Partner shall pilot, when applicable, system functionality prior to Statewide rollout in accordance with the Business Process Transition Plan. (CDL TM 056)	
TM 13.2.14 For Phase I, the Business Partner shall verify the local welfare system and auditor controller interface connections. For Phase II, the Business Partner shall verify connections between the CCSAS CSE system's standard interfaces and the local welfare system and county courts.	The State will provide staff knowledgeable in local IV-A and County Courts interfaces in order to participate in testing and verifying the local interface connections and to resolve any interagency conflicts. The State will provide any modifications to the CASES application that are necessary to support local interfaces.

TM 13.2.15 For Phase I, the Business Partner shall set up CASES to run in the LCSA-specific environment. The Business Partner shall establish the CASES security table and enter county-specific parameters. The Business Partner shall perform CASES connectivity set-up and forms configuration.	
TM 13.3 User Training	
TM 13.3.1 The Business Partner shall provide CASES training management and coordination, including: 1. Set up and maintain the CASES training environment, 2. Configure the existing PCs and printers in the CASES training room to work with the CASES training environment 3. Set up the CASES training application, batch, and online environments 4. Populate and refresh the CASES training databases 5. Coordinate CASES training activities with the CSE Phase I implementation schedule 6. Monitor the CASES training schedule and progress 7. Develop and monitor the implementation of a CASES Training Plan (CDL TM 098) The Business Partner shall develop and conduct a training program for CSE Version 1 Statewide Services in accordance with the User Training Plan (CDL TM 064-1) The Business Partner shall develop and conduct a training program for CSE Version 2 to train users on how to use the CSE application in accordance with the User Training Plan (CDL TM 064-2).	The State will be responsible for providing training hardware/software environments to support CASES training during Phase I. The State will conduct the training delivery for the CASES application, including arranging for training facilities. All State trainees will have a basic knowledge of how to perform their job functions before training, including child support policy and procedures before attending CSE Version 2 training. The State will be responsible for training new workers during CSE roll-out. The State will be responsible for State trainees having a basic working knowledge of Windows desktop environments and Internet browsers prior to training. The State will be responsible for the training of SDU customer service representatives on CSE. The State is responsible for leading, managing, and verifying that CASES training is being conducted by the CASES Consortium and the LCSAs. The State will continue the existing Train the Trainer approach to delivery, training key LCSA staff as trainers who then become responsible for training the remaining staff in their LCSAs. The State will provide an LCSA training liaison in each LCSA to work with the Business Partner to identify and schedule users for the appropriate training program The State will be responsible for distributing scheduling information to the LCSA staff and confirming their time away from normal business duties to attend training.

TM 13.3.1.V1 (added per CR-00124a) **Training Material Development** The Business Partner shall deliver Training for the Trainer (T4T) training materials utilizing scenario-based application simulations following the "Show Me/Let Me" learning methodology. The Business Partner shall develop sufficient "Show Me" scenarios for read-only online functions to enable the Trainers to fully train the end users. The Business Partner shall develop "Show Me" scenarios and "Let Me" scenarios for those online update functions with unique functional paths. Those paths with repetitive transaction types may be limited to a few representative examples. The Business Partner will provide the required software to develop simulations. The Business Partner will develop training simulations in the Version 1 System Test environment using screens that have passed testing. The training simulations and user procedures will be tested following the current approach outlined in the Version 1 User Training Plan (CDL TM 064-1). These materials will be used to develop Word document based user procedures, quick reference materials, and "cheat sheets". T4T Training modules and user procedures will be posted and accessed via the Project Website.	**Training Material Development** The State will provide resources to help design, develop, and test the T4T training materials at the same level as previously agreed upon in CR-00087.
Training Material Delivery (added per CR-00124a) The Business Partner shall provide Training for the Trainer (T4T) training for CSE Version 1 prior to CSE Version 1 operation. The T4T training shall: • Be held at four (4) regional sites to be selected by the State. • Provide not less than eight (8) T4T sessions which are no longer than two (2) days in duration. • Be designed to support up to 24 students and two (2) Business Partner Staff. The Business Partner will provide instructors for one class of DCSS trainers after the completion of the LCSA classes. The Business Partner will support State staff by providing training developers to answer T4T participant questions via an email hotmail account.	**Training Material Delivery** The State shall provide four (4) regional sites for the Business Partner T4T training program. The State shall provide a total of twenty six (26) desks for twenty four (24) students and for two (2) instructors. These training facilities will contain the same specifications as the Business Partner provided for the Version 2 Training facilities. The State will be responsible for assessing and providing for any special hardware/software needs, such as screen reader software. The State will identify, schedule and notify participants. The State will provide the PCs in the regional training facilities with the following specifications: • Operating System – Windows XP

	Professional with Service Pack 2 (SP2); • Internet Browser – Internet Explorer 5.0 or higher; • Java Virtual Machine – 5.0.0.3085 or higher. T4T attendees will use the T4T Training Materials to provide training to the LCSA's. The State shall verify that the classrooms are ready for training. This includes any verification of hardware, software and connectivity. The State will resolve infrastructure (hardware, software, connectivity or facilities) issues that occur during training excluding any infrastructure provided by the Business Partner.
TM 13.3.2 The Business Partner shall produce a Train the Trainers Plan for CSE Version 2 and shall conduct a training program for trainers in accordance with the Train the Trainers Plan. (CDL TM 062)	The State will identify and make available 240 staff for a period of three weeks to participate in the Train the Trainer sessions in the Sacramento Regional Training Center. The State and LCSA will provide designated staff, who attended T4T, as coaches during CAT session in their local offices.
TM 13.3.3 The Business Partner shall conduct training transition activities in accordance with the Training Transition Plan. (CDL TM 063) The Business Partner shall conduct CSE Version 2 training activities from inception until October 1, 2008 (after V2.3 training materials have been provided by the BP) and shall complete transition of training activities and responsibilities to the State at that time. The Business Partner shall maintain and operate the CSE training program in accordance with the User Training Plan (TM 064). **(updated per CR-C-01157)**	The State will accept full responsibility for the training program commencing October 1, 2008.
TM 13.3.4 The Business Partner shall produce a User Training Plan (CDL TM 064-1, TM 064-2).	

TM 13.3.5 The Business Partner shall produce training materials that reflect the contents and behavior of software releases. Training materials shall be versioned and shall include document version numbers, training material effective dates and the corresponding software release version number.	The State is responsible for developing and maintaining the CASES training materials.
TM 13.3.6 Deleted.	
TM 13.3.7 The Business Partner shall conduct CSE Version 2 end-user training classes at the county and regional level.	
TM 13.3.8 The Business Partner shall provide a method for the State to provide feedback on the effectiveness of the CSE Version 2 training program.	The State will provide input to the CSE Version 2 post training effectiveness survey.
TM 13.3.9 The Business Partner shall establish specific, measurable criteria for assessing the effectiveness of the CSE Version 2 training program, and shall modify the program to improve areas of weakness.	The State will provide input to the CSE Version 2 post-training effectiveness survey and input and approval on modifications to the training program.
TM 13.3.10 The Business Partner shall track attendance at CSE Version 2 training sessions and shall provide the State access to the training attendance records in accordance with the User Training Plan. (CDL TM 064-2)	The State will provide input to and approval of the training attendance tracking method.
TM 13.3.11 The Business Partner shall update the CSE Version 2 training environment with changes that occur in production in accordance with the System Configuration Management Plan. (CDL TM 003)	The State will update training materials and the training environment as application changes are made to CASES.
TM 13.3.12 The Business Partner shall establish a CSE Version 2 training environment with response times representative of production performance levels.	
TM 13.3.13 The Business Partner shall populate CSE Version 2 required training tables, including case information, user profiles, and user IDs.	The State will populate CASES required training tables, including case information, user profiles, and user Ids. The State will provide information on users to be trained and the required training.
TM 13.3.14 The Business Partner shall make the CSE Version 2 training environment available from 7am to 7pm PT Monday through Friday.	
TM 13.4 Hardware and Software Installation	
TM 13.4.1 The Business Partner shall install the hardware and software in accordance with the	The State will provide the WAN / LAN infrastructure and access for Phase I.

Hardware and Software Installation Plan. (CDL TM 067)	The State will identify a specific address and contact name for each county site to ship network and hardware components for storage. The State will provide a primary, secondary, and tertiary contact list, including business, pager, and cell numbers, for hardware installation. Each site must also provide a secure environment and staging area to store network and hardware components until they are ready for installation. The State will provide access to hardware installation sites, free from blocked physical access and/or firebreaks. The State will provide full conduit when necessary for cabling installation. The State will provide 24 hours notice for rescheduling of cabling installation. The State will be responsible for hazardous materials removal if necessary. The State will support the verification of site readiness for hardware and software installation activities, and coordinate activities to make sites ready in accordance with the Hardware and Software Installation Plan. (TM 067) The State will resolve site-specific issues related to installation and escalate problems as necessary. The State will be responsible for de-installation of existing equipment in accordance with the Hardware and Software Installation Plan. (CDL TM 067)
TM 13.4.2 The Business Partner shall record the current version of the application software in accordance with the Software Version Description (SVD). (CDL TM 068)	
TM 13.4.3 Deleted.	
TM 13.4.4 The Business Partner shall produce and distribute Release Notes. (CDL TM 070)	
TM 13.4.5 The Business Partner shall install and check-out the executable software for each installation. Every six months, commencing with the first software delivery, the Business Partner shall demonstrate to the State that the current version of the CSE application software can be regenerated.	
TM 13.4.6 The Business Partner shall support the	The State will be responsible for installation

State and LCSAs in Phase I installation contingency planning. The Business Partner shall produce an Installation Contingency Plan (CDL TM 071) for Version 2.	contingency planning for the 14 counties that transition to CASES in Phase I. The State will provide input to the plan based on site-specific requirements, and will be responsible for executing the plan as necessary. The State is responsible for de-installation, packing, transportation and storage of equipment in accordance with the Installation Contingency Plan. (CDL TM 071)
TM 13.4.7 The Business Partner shall assist the State in establishing CASES hardware / software environments for production and data conversion of the 14 counties that transition from KIDZ, and STAR/KIDS to CASES during Phase I. The Business Partner shall install supporting software, test and confirm the CASES production and data conversion environments, and set-up security access.	For Phase I, the State will provide the hardware / software. 3270 emulation and WAN necessary to maintain the CASES environments. The State will provide the production environment for CASES, and the State will provide operations and support for the production CASES environment
TM 13.4.8 The Business Partner shall plan and conduct pre-installation site survey(s) for each hardware and software installation site. The scope of the survey(s) shall be sufficient to allow for the installation of equipment and software as described in the Hardware/Software Installation Plan (CDL TM 067). The Business Partner shall document the results of each pre-installation site survey. The survey results shall describe any preparatory work that must be completed prior to installation, and identify any inconsistencies, errors or omissions in State provided information that affects the Business Partner's ability to install the equipment and software.	The State will provide the pre-installation site survey information as defined in the Hardware/Software Installation Plan (CDL TM 067) or upon request by the Business Partner. The State will provide or facilitate access to the sites where equipment and/or software will be installed as defined in the Hardware/Software Installation Plan (CDL TM 067) and in accordance with the CCSAS CSE Project Schedule.

TM 14 Production Support	
TM 14.0 Production Support General	
TM 14.0.1 The Business Partner shall maintain required reference data in the CSE System, characterized as static or periodically updated data, such as zip code tables, annual tax tables, cost of living adjustments, Federal Information Process Standards (FIPS), and the Federal Interstate Referral Guide Roster. Reference data also includes code tables, which map codes used by the system to descriptive text. The Business Partner shall also provide the means for the end-user to maintain State and LCSA specific administrative data in the CSE System, such as contact information, employer data, and employee information. Where State and LCSA specific data is provided to the CSE System in a usable format, via an automated interface, the system shall provide automated update capability.	The State will provide reference data to the Business Partner. The State and LCSAs will maintain their specific administrative data, such as employer data, contact information, employee information and addresses.
TM 14.0.2 The Business Partner shall assess Customer Service Support Center(s) needs and produce a Customer Service Support Center Assessment. (CDL TM 072) The Business Partner shall provide the State with the alternatives to consider in selecting the preferred solution.	The State will coordinate Business Partner access to existing LCSA Customer Service Support Centers. The State will arrange for personnel knowledgeable in customer service support to participate in the Customer Service Support Center Assessment. (CDL TM 072) The State will provide the Business Partner with the available processes and procedures from existing LCSA Customer Service Support Centers. The State will provide the Business Partner with the available existing call metrics and VRU scripts for the CCSAS statewide Customer Service Support Center. Within 20 State business days following Business Partner submission of the alternatives, the State will report to the Business Partner the selected solution and the approach that the State requires to be taken with respect to the Customer Service Support Center for input to the Customer Service Support Center Assessment. (CDL TM 072)

TM 14.0.3 The Business Partner shall establish a Customer Service Support Center in accordance with the Customer Service Support Center Plan (CDL TM 073) (updated per CR-2-00247).	The State will assume the responsibilities outlined in the transition section of the Customer Service Support Center Plan (CDL TM 073) and will assume the responsibility for functions and infrastructure as outlined in the Customer Service Support Center Plan (CDL TM 073).
TM 14.0.4 The Business Partner shall provide management, oversight, and technical support staff for the Customer Service Support Centers' operations, from inception of the Customer Service Support Centers through October 2008. The Business Partner shall transition technical support operations to the State in October 2008 (updated per CR-2-00247)	The State will participate and assist in the management, oversight, and technical support of the regional Customer Service Support Centers. The State will assume the responsibilities outlined in the transition section of the Customer Service Support Center Plan (CDL TM 073) and will assume the responsibility for functions and infrastructure as outlined in the Customer Service Support Center Plan (CDL TM 073).
TM 14.0.5 Deleted.	
TM 14.0.6 The Business Partner shall provide Customer Service Support Center operations training at each regional site, and shall produce Customer Service Support Center Training Materials. (CDL TM 074) The Business Partner shall be responsible for Customer Service Support Center operations training from inception to one year following full implementation of the CCSAS CSE system, and shall transition training to the State at that time. The training shall include CSE System user training in accordance with the User Training Plan. (CDL TM 064-2) The Business Partner shall supply on-going training at the rate of four hours per month per Customer Service Support Center staff.	The State will actively participate in, and work jointly with the Business Partner on, the development of the Customer Service Support Center Training Materials. (CDL TM 074) The State will maintain the training materials after the deliverable has reached final approval. After the initial training courses are conducted, the State trainers will conduct on-going training activities with one Business Partner provided trainer. The State is responsible for the training activities after the Customer Service Support Center is transitioned to the State.
TM 14.0.7 The Business Partner shall provide the Customer Service Support Center staff the training necessary to perform the duties and tasks assigned to Customer Service Support Center staff to receive and respond to customer inquiries and questions in accordance with the Customer Service Support Center Plan (CDL TM 073) The Business Partner shall participate in the development of Customer Service Support Center staff training on the Child Support Program necessary for Customer Service Support Center staff to perform.	The State will develop and lead the Child Support Program portion of the training for the Customer Service Support Center staff.
TM 14.1 Operations	

TM 14.1.1 The Business Partner shall implement and operate the CSE system in a manner and location to enable California state employees to manage and oversee the system.	The State will provide 24-hour advance notice, when possible, to the Business Partner to gain access to the processing site(s). The Business Partner will also accommodate access requests without notice for management and oversight purposes. Visits to Business Partner sites, including the Business Partner Service Delivery Center, by State will require an escort at all times.
TM 14.1.2 The Business Partner shall provide full operations responsibilities for Version 1 statewide services commencing with the first release into production. The Business Partner shall commence operations responsibilities concurrent with the first release CSE into production and shall maintain and operate the system until October 31, 2010. (updated per CR-C-01157)	
TM 14.1.3 The Business Partner shall produce a System Operation Manual. (CDL TM 075)	
TM 14.1.4 The Business Partner shall conduct service delivery in accordance with the Service Delivery Management Plan. (CDL TM 076)	
TM 14.1.5 The Business Partner shall gather information in accordance with the Service Delivery Management Plan (CDL TM 076) The Business Partner shall report service level attainment in accordance with the Service Delivery Management Plan. (CDL TM 076)	
TM 14.1.6 The Business Partner shall produce and execute a System Security Plan. (CDL TM 078)	The State will designate a security person to assist the Business Partner with security incident activities, the investigation of security incidents, and actions that result from the investigation of security incidents that may affect State, DCSS, LCSA or state staff.
TM 14.1.7 Deleted.	

TM 14.1.8 The Business Partner shall … *conduct risk analysis* every six months beginning six months after State acceptance of the System Security Plan (CDL TM 078) *(ACF H-1a).* The risk analysis shall …*measure the system's vulnerability to fraud or theft, loss of data, physical destruction, unauthorized access, intrusion, and harm to agency activities (ACF H-1b).* The Business Partner shall perform …*special evaluations whenever a significant change to the system's physical security, hardware or operating system software occurs (ACF H-1c).* The Business Partner shall document the results of the risk analysis in a System Security Assessment Report (CDL TM 079).	
TM 14.1.9 The Business Partner shall develop and implement …*procedures for system and terminal user identification assignment, maintenance, and cancellation.* (CDL TM 080) *Delegation and maintenance of the password administration must be limited to a select number of people, and a mechanism must be in place to quickly notify those responsible when there are personnel changes (ACF H-2e).*	The State will develop procedures in support of system and terminal user identification assignment and maintenance.
TM 14.1.10 The Business Partner shall report security incidents and the results of security investigations in accordance with the System Security Plan. (CDL TM 078)	
TM 14.1.11 Deleted.	
TM 14.1.12 The Business Partner shall produce Local Desk Procedures for System Administration and Network Administration. (CDL TM 083)	**TM083 & TM084 combined per CR-2-00295**
TM 14.1.13-(combined with TM 14.1.12 per CR-2-00295)	
TM 14.1.14 The Business Partner shall produce a Software User Manual Plan (SUMP). (CDL TM 085)	

TM 14.1.15 The Business Partner shall produce and execute a Performance and Capacity Management Plan. (CDL TM 086-1, TM 086-2) The performance and capacity metrics contained within the Performance and Capacity Management Plan shall be based on the related technical Specifications. The performance and capacity management metrics for Version 1 defined in the Performance and Capacity Management Plan (CDL TM 086-1) may provide an optimization and tuning period to allow a progressive approach for reaching the Version 1 production performance objectives. The duration of the Version 1 optimization and tuning period shall be no more than 3 months starting from the date of Version 1 implementation.	
TM 14.1.16 The Business Partner shall monitor and report on system performance and availability in accordance with the Performance and Capacity Management Plan. (CDL TM 086-1, TM 086-2)	
TM 14.1.17 The Business Partner shall provide the tools at the server, client, database, application, interface and internet component levels to measure areas of system and network performance.	
TM 14.1.18 The Business Partner shall report on system utilization in accordance with the Performance and Capacity Management Plan. (CDL TM 086-1, TM 086-2)	
TM 14.1.19 The Business Partner shall use performance-monitoring tools that are configured to assist in prediction of system failures.	
TM 14.1.20 The Business Partner shall provide on-site hardware maintenance support for hardware that the Business Partner supplies.	
TM 14.1.21 The Business Partner shall maintain a State approved list… *of retention period for all application and operating system files and programs version (ACF H-5b) in a Retention Control Document and deliver the current copy to the State project records center.*_ (CDL TM 089)	The State will assist with the identification and update of the state-approved list of retention periods and provide the Business Partner with the state-specific retention period requirements.
TM 14.1.22 The Business Partner shall …*conduct routine, periodic backups of all child support enforcement system files, data files, application programs and documentation (ACF H-5e)* in accordance with the System Operation Manual. (CDL TM 075)	

TM 14.1.23 The Business Partner shall store back-ups of the new statewide system …*files, programs, documentation, data files, etc. off-site in secure, waterproof,* earthquake proof, *and fireproof facilities (ACF h-5f).*	
TM 14.1.24 The Business Partner shall …*retain off-site a* four years plus four months *history of the new statewide system database in a form retrievable through automated system recovery and restore procedures (ACF H-5c).* This history shall be extended to seven years for IRS-related information.	
TM 14.1.25 The Business Partner shall, in accordance with the Database Management Plan (CDL TM 025), provide the capability to:	
TM 14.1.25.1 Recover or restore CSE databases associated with the CSE environments to a consistent and usable database state in the event of either a natural or technical disaster.	
TM 14.1.25.2 Backup and recover interdependent databases.	
TM 14.1.25.3 Recover and restart systems operation after outages caused by system crashes (e.g., power outages) or server crashes.	
TM 14.1.25.4 Isolate suspect portions of its physical database structure while bringing the rest of the database online for (limited) use.	
TM 14.1.25.5 Implement a database backup strategy that supports restore from the previous full backup plus changes logged since the previous full backup.	
TM 14.1.25.6 Conduct backups of the CSE DBMS database objects without exclusion (e.g. user defined items, related indexing information, etc.).	
TM 14.1.25.7 (Deleted)	
TM 14.1.25.8 Use commercially available backup media libraries (e.g., tape silo).	
TM 14.1.25.9 Conduct unattended backup and restore operations for data in the CSE environments.	
TM 14.1.26 The Business Partner shall, on a regular basis, make backup copies of DBMS system tables such that this information is maintained in a form suitable for restoration..	

TM.14.1.27 The Business Partner shall provide an …*approved Disaster Recovery Plan* (CDL TM 090) *which provides detailed actions to be taken in the event of a natural disaster (fire, water damage, etc.) or a disaster resulting from negligence, sabotage, move action, etc. (ACF H-5a).* The Business Partner shall conduct all disaster recovery activities, in the event of a disaster, in accordance with the Disaster Recovery Plan. As part of the Disaster Recovery Plan. (CDL TM 090) the Business Partner shall provide: • A Disaster Recovery time not to exceed 48 hours for the critical CSE applications. A subset of the production environment (transaction and batch processing (without image support) and the ability to send out payments) shall be restored in the event of a declared disaster. • A 96-hour Disaster Recovery solution for other production components, including the image component and data warehousing. • The time taken to restore the image date in the event of a disaster is subject to the quantity of data and the achievable transfer rate, and is not guaranteed to be complete within 96 hours (bullet added per CR-2-01020).	
TM 14.1.28 - The Business Partner shall establish a back-up facility and shall exercise the Disaster Recovery Plan (CDL TM 090) at the back-up facility semi-annually, commencing with the first production implementation.	The State will participate in the testing as described in the Disaster Recovery Plan.
TM 14.2 Maintenance	
TM 14.2.1 The Business Partner shall plan and conduct software system maintenance in accordance with the System Operation Manual. (CDL TM 075)	The State will identify State, DCSS or LCSA dependencies that must be integrated into the System Maintenance Plan (CDL TM 091) and will notify interested parties of maintenance activities.
TM 14.2.2 The Business Partner shall report on system maintenance in accordance with the System Operation Manual. (CDL TM 075)	

TM 14.2.3 The Business Partner shall produce and execute a Software Transition Plan (STrP). (CDL TM 093)	The State will assign a Software Transition Coordinator that will be responsible for problem resolution activities during the transition period. The State will perform the activities required to transition the responsibility for software maintenance and operations to the State. These activities will include: ▪ Provide staff for training activities performed by the Business Partner; ▪ Identification of the new processing location and access to the technical staff supporting the transfer of services; ▪ Identify organizations responsible for application maintenance, application enhancement, help desk, operations, and technical support and facilitate the transition of those services.
TM 14.3 Help Desk	
TM 14.3.1 The Business Partner shall produce and execute a Help Desk Plan. (CDL TM 094)	The State will provide information (including processes, procedures, description of physical infrastructure, and access to a limited number of help desk staff) to the Business Partner to facilitate development of the Help Desk Plan.
TM 14.3.2 The Business Partner shall use a Help Desk tracking tool that employs access control mechanisms.	
TM 14.3.3 The Business Partner shall provide access to the Help Desk tracking tool to the State for search, direct query, custom reporting and export capability for creation of metrics and analysis of trends.	
TM 14.3.4 The Business Partner shall answer Help Desk calls during the hours of 6am to 6pm PT, Monday through Saturday with the exception of: New Year's Day, Christmas Day, Fourth of July, Thanksgiving, Memorial Day and Labor Day.	The State will accept inquiries for items that are outside of the Business Partner's scope of work.
TM 14.3.5 The Business Partner shall utilize an automated call directing system that allows for call distribution among Help Desk staff and reports, by individual and collectively, on call, and on-call metrics such as hold time, talk time, and abandoned calls.	

TM 14.3.6 The Business Partner shall train appropriate State staff in the use of the Help Desk's automated call directing system and problem tracking system to designated staff prior to initial use.	
TM 14.3.7 The Business Partner shall provide access to the Help Desk's problem tracking system as defined in the Help Desk Plan. (CDL TM 094)	
TM 14.3.8 The Business Partner shall provide and maintain a web site for frequently asked questions (FAQs) accessible by DCSS, LCSAs and CCSAS Project staff. At a minimum, the web site shall contain information about Help Desk support including hours of operation, available services, methods of contact, system status, announcements, virus alerts, service bulletins, CCSAS CSE project timing, training information, and implementation schedules. In addition the web site shall contain sections for self help, user feedback and other frequently asked questions. The website shall be organized so that DCSS, LCSAs and CCSAS Project staff can access answers to commonly asked questions.	
TM 14.3.9 The Business Partner shall provide Help Desk services for all CSE Version 1 users. Level 1 Help Desk services for CSE Version 1 users are based on the following call volumes for the following calendar years: 2005 – 3,540 2006 – 13,320 2007 – 6,120 2008 – 3,240 (added per CR-00124a)	

PART 3
TECHNICAL MANAGEMENT

CONTRACT DELIVERABLE LIST (CDL)

SOW/CDL Traceability. There is bi-directional traceability between the SOW and the CDL. CDL Items are declared within the SOWs, and have an associated Contract Deliverable List (CDL) description. As an example, in the SOW:

PM 10 Quality Management…

PM 10.2 The Business Partner shall produce a Quality Management Activity Report. (**CDL PM 014**)

The CDL Item descriptions provide a cross-reference to the SOWs that reference the deliverable in the field named SOW Paragraph Reference. In our example, CDL PM 014, Field 10 SOW Paragraph Reference = PM 10.2.

Table 1 provides the definition for each field contained within the CDL Item description.

Table 1 Contract Deliverable List Field Definitions

FIELD NAME	FIELD DESCRIPTION
1) Deliverable Identifier:	Unique identifier assigned to the CDL Item. This identifier is specified within each Statement of Work that references the deliverable.
2) Deliverable Name:	Name of the CDL Item
3) Standard:	Standard for the deliverable format / content. If no standard is indicated, the Business Partner shall recommend a format for the deliverable, which shall require State acceptance.
4) Acceptance:	The conditions under which the State will accept a CDL Item. There are three levels of acceptance: 1) Walk-thru prior to delivery – State review and acceptance required 2) State review and acceptance required 3) State acceptance required
5) Timeframe for State acceptance:	Number of days allocated to the State for review and acceptance following formal deliverable submission for acceptance.
6) Frequency of Submission:	Reserved for regularly occurring submissions: weekly, monthly, etc. If this item is completed, then Date of Subsequent Submission is not applicable.

FIELD NAME	FIELD DESCRIPTION
7) Date of First Submission:	Date the deliverable shall be initially due. The dates are expressed as relative dates, tied to a project milestone such as contract award. Note that in some instances the submission date is the milestone plus a stated period of time (e.g. Project Start plus 1 month), and in other cases the due date precedes the milestone (e.g. System Requirements Review minus 3 months). For all deliverables that require a walk-thru prior to delivery, the draft deliverable shall be made available a minimum of 10 State business days prior to the scheduled walk-thru. The Date of First Submission reflects the date that the reviewed and revised document shall be submitted for acceptance, which shall be no more than 10 State business days following the walk-thru.
8) Date of Subsequent Submission:	Indicates when non-regularly-scheduled (Frequency of Submission is N/A) deliverables shall be resubmitted. If this item is completed, then Frequency of Submission is not applicable. The Date of Subsequent Submission may be designated 'As Necessary and mutually agreed upon by CCSAS project staff and the Business Partner,' and would be invoked when it is agreed that an event warrants that the CDL Item be updated.
9) Delivery / Format:	Delivery / Format for the deliverable. The options include: 1) Electronic / In Accordance with CCSAS Office Automation Standards: 2) Hard Copy / In Accordance with CCSAS Office Automation Standards 3) Electronic & Hard Copy / In Accordance with CCSAS Office Automation Standards
10) SOW Paragraph Reference:	Cross-references CDL Item to the Statement Of Work paragraphs in which the deliverable is referenced.
11) Content:	May contain tailoring instructions for the cited standard, or Content outline / description for those CDL Items where the Standard is Business Partner format with State acceptance.

1. DELIVERABLE IDENTIFIER TM 001

2. DELIVERABLE NAME System Engineering Management Plan (SEMP)

3. STANDARD IEEE 1220 - Annex B

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION First System Requirements Review minus 3 months

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 0.3, TM 8.3

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 002

2. DELIVERABLE NAME Software Development Plan (SDP)

3. STANDARD J-STD-016-1995 E.2.1

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 3/15/04 plus or minus 5 State Business days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 0.5, TM 12.5.2, TM 12.6.13, TM 12.6.14, TM 12.9.14

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

Subclause 4.1 shall be interpreted to include the following text:

Subclause 4.1 shall specify the process(es) and methodologies for:

a. Avoidance and clearance of significant defects, as defined by the Problem Resolution Management Plan (CDL TM 010)
b. Assigning defect corrections to periodically scheduled releases, e.g. quarterly releases
c. Scheduling releases due to the clearance of significant defects
d. Preparing Release Schedules for both periodically scheduled releases and releases due to the clearance of significant defects

Subclause 4.1 shall identify the roles and responsibilities for the Business Partner and State in establishing the Release Schedule with the State making the final determination regarding the schedule. Release Schedule(s) shall be prepared for both periodically scheduled releases and releases due to the clearance of significant defects.

1. DELIVERABLE IDENTIFIER TM 003 (updated per CR-00083-1)

2. DELIVERABLE NAME System Configuration Management Plan

3. STANDARD IEEE 828-1998

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Project Start + 1 month

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 4.2, TM 4.6, TM 12.6.5, 13.3.11

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

The content of the System Configuration Management Plan will now include reference to System Configuration Records (formerly CDL TM005) which were changed to a Work Product with CR00083-1. The Work Product Records will be produced monthly, maintained for the life of the contract, and provided to the State project records center.

1. DELIVERABLE IDENTIFIER TM 004
2. DELIVERABLE NAME System Configuration Management Procedures
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION 12/30/03 (updated per CR-00025)
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 4.5
11. CONTENT
The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction
This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose
The purpose of Configuration Management Procedures is to establish a repeatable series of steps for performing activities that comprise the configuration management process. This document defines the procedures for performing configuration these activities. These procedures can then be communicated to Business Partner and state staff participating in the configuration management process so that requests are submitted, processed, tracked, and distributed in a consistent and controlled manner. Instituting configuration management procedures across Business Partner and CCSAS project staff supports the integrity of configuration items and reduces rework associated with changes being introduced haphazardly. Configuration Management Procedures will be done in accordance with ACF H-3b requirements.

1.2 Document Overview
This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions
This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents
This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Requesting Changes
This section shall describe the steps for requesting a change to a configuration item. This section also defines who will be authorized to submit change requests and the information a request must contain in order to be considered.

3. Processing Change Requests
This section shall describe the steps that will be followed in evaluating, approving/rejecting, and communicating the disposition of a configuration item change request. This section also defines the general criteria and procedure for escalating a configuration item change request to the Scope Control Management Process.

4. Implementing Changes
This section shall define the procedures and steps to program/institute, and implement approved changes into the CSE and/or CCSAS system. As part of change implementation, this section shall describe programming, testing and validation standards; such as verify and validate changes to the master files and application software. Implementation of changes will be done in accordance with ACF H-3a.

5. Tracking Changes
This section shall define the procedures and steps to track configuration item changes upon configuration item change request approval.

6. Distributing Changes and Release Management
This section shall describe the steps for distributing/migrating configuration items between environments after verification and validation approval that a change has passed testing and implementation requirements. It shall include release management procedures for both periodically scheduled releases (e.g. quarterly) and release due to the clearance of significant defects. This section will also define the criteria that must be met prior to distributing a change into the testing and production environments and who is authorized to distribute changes.

7. Maintaining Past Versions
This section shall describe the procedures and use of version control software to maintain past versions of configuration items and to control versions of configuration items.

8. Configuration Audits
This section shall describe the procedures for conducting the configuration audits specified in the System Configuration Management Plan.

9. Other Procedures
This section shall describe procedures for performing other configuration management activities not covered by sections 2-8 of this deliverable standard.

1. DELIVERABLE IDENTIFIER TM 005 – Converted to Work Product with CR00083-1

2. DELIVERABLE NAME System Configuration Records

3. STANDARD

4. ACCEPTANCE No formal State acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE N/A

6. FREQUENCY OF SUBMISSION Monthly

7. DATE OF FIRST SUBMISSION Beginning with the month of January 2004 with a submission date of 2/17/04 (updated per CR-00025)

8. DATE OF SUBSEQUENT SUBMISSION The 10th State Business day after the last Friday of each month (updated per CR-00025).

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 4.6

11. CONTENT

CR00083-1 changes this CDL to a Work Product. Information required in this CDL will now be included in CDL TM003.

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Configuration Record contains information captured in the problem resolution and version management tools for a build or configuration item. A template for this deliverable will be created to provide configuration management staff with a reference for extracting the configuration information needed to produce a configuration record using the reporting capabilities of the problem resolution and version management tools.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Configuration Records

The Business Partner will leverage the reporting capabilities of the version and problem resolution management tools to capture and report the configuration information required by the Configuration Records deliverable. This section shall describe the configuration

information available from the problem resolution and version management tools. Subsections shall describe available data/reports to include the following:

- Build Data and Reporting
- Configuration Item Change Activity and Reporting
- Configuration Item Problem Data and Reporting

2.1 Build Data and Reporting

This subsection shall contain the data captured by the version management tool for a particular build including the configuration items that comprise a given build that will include version numbers, revision and release information of each entity/build.

2.2 Configuration Item Change Activity and Reporting

This subsection shall contain the data captured by the version management tool for a particular configuration item including a history of the changes made to a configuration item since being placed under project-level or higher configuration control.

2.3 Configuration Item Defect Data and Reporting

This subsection shall contain the defect data captured by the problem resolution management tool for configuration items including the status of reported problems or changes affecting a particular configuration item.

1. DELIVERABLE IDENTIFIER TM 006

2. DELIVERABLE NAME Requirements Management Plan

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Submission 1 – October 21, 2003. The state will have 15 State Business days for State Acceptance. The first submission will contain high level concepts for all sections of the Requirements Management Plan, with detailed descriptions for System Requirements through Software Design phases (updated per CR-00030).

8. DATE OF SUBSEQUENT SUBMISSION Submission 2 – March 5, 2004. The state will have 10 State Business days for State Acceptance. The second submission will contain all of the sections of the Requirements Management Plan including additional details on tools and specific details for Detail Design through Testing phases (updated per CR-00030 and CR-00042).

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 5.2

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1,4,1.5, 1.6, 1.7, 1.8 and 1.9) below.

1.1 Description/Purpose

This subsection shall describe the purpose of the Requirements Management Plan deliverable.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions**.**

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Plans and Processes

This subsection shall summarize related plans and processes that affect activities defined within the Requirements Management Plan.

1.6 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

1.7 Milestones

This subsection shall summarize the milestones that relate to the activities described within this document.

1.8 Document Maintenance
This subsection shall describe the frequency of review cycle of this document. The authority to initiate an out-of-cycle review and who is responsible for the update shall also be defined in this subsection.
1.9 Communication
This subsection shall summarize who should be notified upon approval of this document and upon completion of updates.

2. Requirements Overview
This section provides an overview and the relationship of business, system and software requirements. The following subsections will describe the requirements in greater detail.
2.1 Business Requirements
This subsection shall give a detail definition of a business requirement, and how they relate to system and software requirements. It shall also explain the breakdown categorization of a business requirement.
2.2 System Requirements
This subsection shall give a detail definition of a system requirement, and how they relate to business and software requirements. It shall explain the breakdown categorization of a system requirement. It shall also explain the relationship between system requirements and other products delivered as part of the System/Subsystem Specification.
2.3 Software Requirements
This subsection shall give a detail definition of a software requirement, and how they relate to business and software requirements. It shall explain the breakdown categorization of a software requirement. It shall also explain the relationship between software requirements and other products delivered as part of the Software Requirements Specification.

3. The Requirements Management Process
This section provides an overview of the requirements management process using graphics, narrative, or a combination of both. The following subsections will describe the components of the requirements management process in greater detail.
3.1 Submission of New/Changed Requirements
This subsection shall describe the process by which the State may submit new requirements and/or change/delete existing requirements. It shall also describe the process by which the contractor may introduce such changes. It shall highlight the process, procedures, and tools necessary to handle such submissions.
3.2 Criteria for Quality Requirements
This subsection shall describe the attributes of a quality requirement.
3.3 Categorization
This subsection shall list and define the requirement categories deemed necessary for management and tracking purposes.
3.4 Criticality Assessment and Establishment of Timeframes
This subsection shall describe how requirement change requests will be evaluated and prioritized. It shall also describe how requirements management is linked to our Project Management Approach, e.g., scope management and cost/schedule management.
3.5 Operational Concept Impact
This subsection shall describe the impact analysis that shall be performed for those requirement change requests deemed worthy of further consideration after the criticality assessment process. Analysis shall attempt to estimate level of effort required if the request is

approved. Analysis shall also attempt to identify impact on existing requirements, technical issues, schedule, budget, resource, risks and training.

3.6 Traceability

This subsection shall describe how and why lower-level requirements shall be linked to higher-level requirements/project goals. It will discuss the traceability criteria that will be used to validate that a new/changed requirement is related to a stated need, goal, or higher level requirement. Further, this subsection will discuss the traceability of requirements across designs, test conditions, code, and problems.

3.7 Reconciliation

This subsection shall describe the process by which requirements are reviewed for completeness, traceability, inconsistencies, and redundancies, before the final approval decision.

3.8 Approval

This subsection shall outline the specific approval process and define those positions that have the authority to approve changes, additions, and deletions to requirements. It shall describe the need for interim reviews to discuss status of requirements. Approved requirement change request may require a change order.

3.9 Decision History

This subsection shall describe how supporting documentation regarding the approval process is stored for historical reference.

3.10 Baselined Requirements

This subsection shall describe the how, when, and why requirements are baselined. It shall indicate how project scope is limited to mutually agreed-upon baselined requirements.

4. Business Requirements Traceability

This section shall describe the bidirectional traceability of requirements, design, code and testing products. Requirements traceability shall cover both horizontal and vertical relationships. The following subsections shall describe in detail the components of requirements traceability.

4.1 Business Requirements to System Requirements

This subsection shall describe how DOORS provides traceability of business requirements to system requirements.

4.2 System Requirements to Software Requirements

This subsection shall describe how DOORS provides traceability of system requirements to software requirements.

4.3 Requirements to Design Traceability

This subsection shall describe how DOORS provides traceability of requirements to design documents.

4.4 Design to Code Traceability

This subsection shall describe how Rational Rose traces design documents to application source code and discusses the transitive nature of requirements/design/code traceability.

4.5 Requirements to Test Traceability

This subsection shall describe how Mercury Interactive Test Director provides traceability of test cases to requirements.

4.6 Problem to Test Traceability

This subsection shall describe how Mercury Interactive Test Director provides traceability of problems to test cases.

4.7 Problem to Code Traceability

This subsection shall describe how Mercury Interactive Test Director provides traceability of problems to source code modules affected by the problem.

5. Reports

This section shall contain an overview of reporting capabilities related to requirements and change proposal requests.

5.1 Reports on Requirements

This subsection shall summarize the reports available related to requirements to address traceability, quality and other metrics.

5.2 Reports on Change Requests

This subsection shall summarize the reports available related to change requests submitted or approved.

1. DELIVERABLE IDENTIFIER TM 007

2. DELIVERABLE NAME Knowledge Management Plan

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Project start + 3 months

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 7.1

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the three subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Knowledge Management Plan describes how the CCSAS CSE project team and the Business Partner intend to conduct, assess, and report knowledge transfer activities. The Knowledge Management Plan is a user-oriented document that describes the approach, methodology, requirements, and plan for providing skill and knowledge development opportunities to the State staff involved with the CSE application. Because the purpose of the Knowledge Management Plan is to act as the skill and knowledge transfer roadmap, the Knowledge Management Lead uses this plan to design and conduct learning activities. The CCSAS CSE project team and the Business Partner also use this plan to track and evaluate knowledge transfer progress.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions**.**

1.4 Referenced Documents

This section shall list the number, title, revision, date, and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This section shall summarize related deliverables or deliverables that are impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Knowledge Management Overview

This section shall describe the purpose and objectives of the Knowledge Management approach.

3. Knowledge Transfer Methodology

This section shall describe the methodology and approach for conducting learning activities to achieve knowledge-transfer objectives. The section shall describe the Plan's approach, including the creation of Individual Development Plans for State staff involved with the CSE application and skill assessment for each learner. This section shall describe how knowledge-transfer activities will be executed through formal, informal, and on-the-job training. This section shall describe the evaluation process that will gather feedback from the learners about the training courses they have taken.

4. Roles and Responsibilities

This section shall describe the CCSAS CSE project team and Business Partner knowledge management roles and responsibilities. The section shall include the types of project positions, the roles and responsibilities by project phase, and the targeted knowledge and skills for each position by project phase. This section shall describe the approach to ensure that the appropriate number of State staff participate in knowledge transfer activities. This section shall also indicate the number of State project staff assigned to the project positions.

5. Knowledge Transfer Requirements

This section shall describe knowledge transfer requirements and the plan for providing formal, informal, and on-the-job training and learning activities to State staff involved with the CSE application. The section shall describe the learning activities and shall indicate the estimated number of training and learning-activity hours by project position or skill set, according to project position, by project phase. This section shall describe the Business Partner-developed training documentation and standards**.**

This section shall describe the Business Partner approach to updating and adjusting Knowledge Transfer activities throughout the Development Life Cycle to ensure current knowledge transfer needs are met.

6. Curriculum Plan

This section shall describe the curriculum for formal training courses. The curriculum shall include the courses by position, including course content descriptions.

7. Assessment Approach (Metrics)

This section shall describe the Plan's assessment approach for monitoring and reporting on the knowledge transfer progress, as measured against specific criteria, as well as each learner's progress toward achieving the goals in his or her Individual Development Plan(s). This section shall describe the approach, timeline, and roles and responsibilities for conducting and tracking the assessments.

8. Timeline and Milestones

This section shall describe the timeline and milestones for the knowledge management activities.

9. Knowledge Transfer Supporting Documentation and Tools

This section shall describe the Knowledge Transfer tools (video presentations, computer based training, etc.) and the method for providing current documentation of all training material, manuals, etc. This section shall describe the approach to ensure that complete and accurate Knowledge Transfer activities are documented and updated as needed.

1. DELIVERABLE IDENTIFIER TM 008
2. DELIVERABLE NAME Software Development Tools Training Materials
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Prior to the First Software Development Tools Training Session (updated per CR-00044)
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 7.3
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner...

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Software Development Tools Training Materials provide the instructor guidelines and instructional content for training the State CCSAS CSE project team members on the use of software development tools. The Software Development Tools Training Materials is an instructor-oriented document that describes the learning objectives, topics, and content points and presents practice exercises for software development tools. This document will contain the training materials for the various software development tools.

1.2. System overview

This subclause shall briefly state the purpose of the system, the software to which this document applies and the interaction between the two. It shall describe the general nature of the system and software development tools; summarize the history of software development tool, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions**.**

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Training Approach
This section shall describe the training approach implemented for the specified software development tool and the rationale for the training approach. Software development tools training approaches will include formal instructor-led classroom training, formal self-paced training, or informal training.

3. Learning Objectives
This section shall describe the learning objectives of the software development tools training. Learning objectives are the activities or knowledge that the training participant will be able to perform or possess after completing the training course.

4. Training Lesson and Topic Information
This section shall describe the structure of the training content. The section shall outline the lessons and topics (within the lessons). The section shall provide the trainer preparation notes and instructional guides for conducting the training and describe the content points for each topic.

5. Training Exercises
This section shall describe the practice exercises or activities for the participants to perform during training. The section shall describe the scenarios, procedures and outcomes of the exercises, where applicable.

6. Training Materials
This section shall describe the training materials for the software development tools training course. The section shall provide the presentation slides, participant handouts, and job aids for conducting and attending the training.

7. Training Evaluation Form
This section shall provide the evaluation form, customized for each software development tools training. The evaluation form is the mechanism to receive participant feedback on the training course. The evaluation form is developed based on the learning objectives.

8. Software Development Tools Training Logistics
This section shall describe the logistics information including date and location of the training course, the instructor, and training participants who attended the course. This section shall also contain the information resulting from the evaluation feedback forms.

1. DELIVERABLE IDENTIFIER TM 009

2. DELIVERABLE NAME Software Quality Plan

3. STANDARD IEEE 730-1998

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Project start + 6 months

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 9.1

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 010
2. DELIVERABLE NAME Problem Resolution Management Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION First System Verification Test Readiness Review minus 1 month
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 10.1
11. CONTENT
The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction
This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose
The Problem Resolution Management Plan documents the general approach, objectives and activities for the problem resolution management process.

1.2 Document Overview
This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions
This subsection shall refer to a set of common project definitions .

1.4. Referenced Documents
This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables
This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Interrelationships with Related Disciplines
This section shall describe the interrelationships between problem management and the following related disciplines:

- Centralized and local Help Desks
- Change Request Management
- System Configuration Management
- Release Management
- Software Development, Defect Correction and Build Planning

- Capacity/Performance Management

3. Involvement of State Staff
The section shall describe how the Business Partner will include the State in the Problem Management process. The following points shall be addressed:
- Severity levels are appropriately set in accordance with Section 8.1 of Rider D taking into consideration input from both the State and the Business Partner with final determination of severity level to be made by the State
- Responses are within Service Level Objectives
- Actions taken place emphasis on user impacts
- Oversight of external service providers is provided to reduce the risk of miscommunication and faulty hand-offs

4. Problem Management Information
The section shall describe what problem management information will be recorded, and how the contractor will use this information to support problem trend analysis and problem root cause analysis.

The problem resolution tracking mechanism shall include: project name, originator, origin, problem number, problem name, software element or document affected, origination date, category and priority, problem re-open status, description, analyst assigned to the problem, date assigned, date completed, recommended solution, impacts, problem status, approval of solution, follow-up actions, corrector, correction date, version where corrected, correction time, and description of solution implemented.

5. Category of Problems
This section shall address the categorization of problems.

6. Problem Prioritization
This section shall describe how the State and the Business Partner will collaborate to prioritize problem resolution with the State making the final determination regarding priority and scheduling of problem resolution.

7. Problem Escalation
This section shall describe the escalation of problems**.**

1. DELIVERABLE IDENTIFIER TM 011
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 012
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 013
2. DELIVERABLE NAME User Interface Standard
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION 4/29/04 plus or minus 5 State Business Days (updated per CR-00050)
8. DATE OF SUBSEQUENT SUBMISSION 8/27/04 plus or minus 5 State Business Days (updated per CR-00050)
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 12.0.3
11. CONTENT
The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction
This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose
The intent of the User Interface Standard (UIS) is to define the universal design and characteristics of visual displays for the CSE system. The UIS shall address the visual and functional characteristics that are applicable to the technology on which the system is to run.

1.2 Document Overview
This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions
This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents
This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables
This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. General Presentation
This clause shall be divided into the following subclauses to describe the general characteristics, behavior, placement, and color-coding of information on the user interface.

- Windows
- Groups
- Lists and Tables
- Labels and Fields
- Cursors and Pointers

3. Methods for User System Guidance

This clause shall be divided into the following subclauses to describe the methods for user system guidance.

- Navigation
- Task to task
- Screen to screen
- Window to window
- Prompts for input
- Task or request status
- Feedback
- Error management
- On-line help
- Help navigation
- Browse-able and context sensitive help

4. System Menus

This clause shall be divided into the following subclauses to describe the standard for management, organization and presentation of System Menus.

- Menu structures
- Grouping and sequencing
- Navigation
- Option selection and execution
- Presentation

5. User Short-cut command definitions and physical characteristics

This clause shall be divided into the following subclauses to describe short-cut commands and fast-path navigation.

- Timing and availability
- Syntax and structure
- Function keys
- Hot keys
- Command arguments
- Feedback
- Help

6. Object characteristics and behavior

This clause shall be divided into the following subclauses to describe object characteristics and behavior upon user activation.

- Placement
- Appearance
- Feedback
- Input devices
- Pointing and selecting
- Dragging, sizing, scaling and rotation
- Manipulation of text objects and windows (including appropriate use)

7. Data entry

This clause shall be divided into the following subclauses to describe form filling or data entry in which the user fills in, selects entries for, or modifies enterable fields presented by the system.

- Form filling structures
- Layout
- Field, label lengths, alignments
- Input considerations

- Alphanumeric text entry
- Choice entries
- Controls
- Feedback
- Navigation
- Graphics and color coding

1. DELIVERABLE IDENTIFIER TM 014
2. DELIVERABLE NAME Software Product Specification (SPS)
3. STANDARD J-STD-016-1995 I.2.1
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 5 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION First System Qualification Test minus 10 State business days
8. DATE OF SUBSEQUENT SUBMISSION With each subsequent System Qualification Test and ORAR minus 10 State business days
9. DELIVERY/FORMAT Electronic/In Accordance with CSE Project Tools
10. SOW PARAGRAPH REFERENCE TM 12.0.4
11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 016-1

2. DELIVERABLE NAME System/Subsystem Specification (SSS)

3. STANDARD J-STD-016-1995 F.2.2

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION System Requirements Review for Version 1 + 10 State business days

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance with CSE Project Tools

10. SOW PARAGRAPH REFERENCE TM 12.1.2

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 016-2

2. DELIVERABLE NAME System/Subsystem Specification (SSS)

3. STANDARD J-STD-016-1995 F.2.2

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 3/29/04, plus or minus 5 State Business Days

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan (updated per CR-00061)

9. DELIVERY/FORMAT Electronic/In Accordance with CSE Project Tools

10. SOW PARAGRAPH REFERENCE TM 12.1.2

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 017-1

2. DELIVERABLE NAME System/Subsystem Design Description (SSDD)

3. STANDARD J-STD-016-1995 G.2.1

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 4/29/04 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance with CSE project tools

10. SOW PARAGRAPH REFERENCE TM 12.2.1

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 017-2

2. DELIVERABLE NAME System/Subsystem Design Description (SSDD)

3. STANDARD J-STD-016-1995 G.2.1

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 8/27/04 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance with CSE project tools

10. SOW PARAGRAPH REFERENCE TM 12.2.1

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 018

2. DELIVERABLE NAME Technical Infrastructure Description

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Acceptance of Software Design Review for Version 1 + 1 month (updated per CR-00056)

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 12.2.2

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

The Technical Infrastructure Description is detailed infrastructure design with traceability to the SSDD and SSS (for technical requirements).

The Technical Infrastructure Description includes the approach to technical infrastructure planning in the System Engineering Management Plan and/or the Software Development Plan, as appropriate.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Technical Infrastructure Description (CDL TM018) describes the design and implementation of the CSE technical infrastructure. Technical Infrastructure incorporates the hardware and system software that constitutes a system environment. It does not include the developed application code.

This document is intended to function as a living document throughout the life of the project to document and manage changes, both planned and unplanned.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall provide the definition of those terms used in the completed deliverable whose meaning must be derived in context or that may be specific to the CCSAS project effort, CCSAS stakeholders, or the alliance.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents

referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2.Strategy

This section shall define the fundamentals behind the use and implementation of the technical infrastructure. It will document or include references to appropriate system requirements and architectural principles that govern the use of technical infrastructure, and the high-level architectural design contained in the System / Subsystem Design Description (SSDD).

The requirements, principles and high-level design will be used in creating the low-level design contained in this document, and on an on-going basis in the evaluation of change requests that impact the technical infrastructure.

3. Physical Infrastructure Description

This section shall provide a physical description of the full server side environment, including each of the eight environments. This description shall include the following:

- An overview of the physical environment, including servers, LPARs, firewalls, routers, circuits, storage, etc.
- Hardware and software product numbers and version numbers.
- Hardware configuration details such as memory and CPU capacities.
- Software product assignment to hardware items, including number and type of licenses.
- The overlay of the eight environments onto the physical infrastructure.

This section shall represent the physical environment as it will be structured in its ending stable state. During product rollout, it will also be maintained so as to represent the current state of the physical environment.

4. Infrastructure Configuration Description

This section shall provide a description of how the system software is configured to support each environment. It shall provide an overview level description that shall be further defined in the succeeding sections, Application Server Design Description and Database Design Description. The description shall include the following:

- The number of sub-environments in each environment
- Mechanisms for separating sub-environments (e.g. WebSphere virtual hosting, DB2 schemas).
- Environment specific configuration information (e.g. memory, priority, max processes)
- Specific roles and authorities within each environment

5. Application Server Design Description

This section shall specify how the application server environment shall be configured. It shall contain information about the allocation of web servers, caching and load-balancing servers, and application servers. Environment specific information shall be identified.

6. Database Design Description

This section shall specify how the database server environment shall be configured. It shall contain environment wide information for each environment such database and schema allocation, buffer pool sizes, memory and storage allocation. It shall provide infrastructural details to support the logical and physical data model work products.

1. DELIVERABLE IDENTIFIER TM 019
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 020-1

2. DELIVERABLE NAME Software Requirements Specification (SRS)

3. STANDARD J-STD-016-1995 F.2.4

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 5/28/04 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan.

9. DELIVERY/FORMAT Electronic/In Accordance With CSE Project Tools

10. SOW PARAGRAPH REFERENCE TM 12.3.1

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 020-2

2. DELIVERABLE NAME Software Requirements Specification (SRS)

3. STANDARD J-STD-016-1995 F.2.4

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 11/16/2004 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance With CSE Project Tools

10. SOW PARAGRAPH REFERENCE TM 12.3.1

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 021-1

2. DELIVERABLE NAME Software Design Description (SDD)

3. STANDARD J-STD-016-1995 G.2.4

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 7/29/2004 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance With CSE Project Tools

10. SOW PARAGRAPH REFERENCE TM 12.4.1, TM 12.6.2, TM 12.6.11.1

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 021-2

2. DELIVERABLE NAME Software Design Description (SDD)

3. STANDARD J-STD-016-1995 G.2.4

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 4/29/2005 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance With CSE Project Tools

10. SOW PARAGRAPH REFERENCE TM 12.4.1, TM 12.6.2, TM 12.6.11.1

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 022
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 023
2. DELIVERABLE NAME Database Development Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION 3/3/2004 plus or minus 5 State Business Days (updated per CR-00050)
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 12.6.1
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Database Development Plan provides a description of the activities used during design and development of the CCSAS CSE database. It includes a discussion of general and detailed database development activities, information on the schedule of database activities, and the organization and personnel involved with database design and build activities.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions .

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Plans for performing general database development activities

This clause shall be divided into the following subclauses. Provisions corresponding to non-required activities may be satisfied by the words “Not applicable.”

2.1 Database development process

This subclause shall describe the database development process to be used. The planning

shall identify planned builds, if applicable, their objectives, and the database development activities to be performed in each build.

2.2 General plans for database development

This subclause shall be divided into the following to describe the approach to be followed for planning and oversight of database design and build activities.

- Database development methods This subclause shall describe or reference the standards and practices to be followed for representing design and build of the database.
- Standards and practices for database use This subclause shall describe or reference the standards and practices to be followed for representing database models, building and maintaining the database, and other activities related to database design and build.
- Traceability This subclause shall describe the approach to be followed for documenting the business need and/or requirement for database entities and elements.
- Recording rationale This subclause shall describe the approach to be followed for recording rationale that will be useful to the maintenance organization for key decisions made on the project. It shall interpret the term “key decisions” for the project and state where the rationale are to be recorded.

3. Plans for performing detailed database development activities

This clause shall be divided into the following subclauses. Provisions corresponding to non-required activities may be satisfied by the words “Not applicable.” The discussion of each activity shall include the approach (methods/procedures/tools) to be applied to: the analysis or other technical tasks involved, the recording of results, and the preparation of associated deliverables, if applicable.

3.1 Planning and oversight

This subclause shall be divided into the following to describe the approach to be followed for planning and oversight of database design and build activities.

- Database development planning (covering updates to this plan)
- Database test planning
- Following and updating plans, including the intervals for management review

3.2 Establishing database development environments

This subclause shall be divided into the following to describe the approach to be followed for establishing, controlling, and maintaining database development environments.

- Database environment(s)
- Database language development library
- Database development files

3.3 Database configuration management

This subclause shall be divided into the following to describe the approach to be followed for database configuration management.

- Preparing source files
- Preparing the “as-built” database design and maintenance information
- Configuration identification
- Packaging, storage, handling, and delivery
- Updating the database requirement specification

3.4 Problem Resolution

This subclause shall be divided into the following to describe the approach to be followed for problem resolution.

- Problem/change reports (with items to be recorded)
- Problem resolution tool

3.5 Joint technical and management reviews

This subclause shall be divided into the following to describe the approach to be followed for joint technical and management reviews.

- Joint technical reviews, including a proposed set of reviews
- Joint management reviews, including a proposed set of reviews

3.6 Risk management

This subclause shall describe the approach to be followed for risk management.

3.7 Coordinating with associate developers

This subclause shall describe the approach to be followed for coordinating activities on the database design and build.

3.8 Process improvement

This subclause shall describe the approach to be followed for process improvement related to database design and build.

4. Schedules

This clause shall present schedule(s) identifying the activities, showing initiation of each activity, availability of draft and final deliverables and other milestones, and completion of each activity.

5. Project organization and resources

This clause shall be divided into the following subclauses to describe the project organization and resources to be applied as they relate to database development.

5.1 Project organization

This subclause shall describe the organizational structure to be used on the database design and build portion of the project, including the authority and responsibility of each organization for carrying out required activities.

5.2 Project resources

This subclause shall describe the resources to be applied to the database design and build portion of the project. It shall include, as applicable:

- Personnel resources, including:
  - The estimated staff-loading for the project (number of personnel over time)
  - The breakdown of the staff-loading numbers by responsibility
  - A breakdown of the skill levels for personnel performing each responsibility
- Overview of facilities to be used, including geographic locations in which the work will be performed, facilities to be used, and secure areas and other features of the facilities as applicable to the contracted effort.

6. Notes

This clause shall contain any general information that aids in understanding this document.

1. DELIVERABLE IDENTIFIER TM 024
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 025 (Deleted per CR-2-00113-01)
2. DELIVERABLE NAME
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

This CDL deleted per CR-2-00113-01, approved on August 13, 2004

1. DELIVERABLE IDENTIFIER TM 026
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 027
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 028
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 029
2. DELIVERABLE NAME SDU Integration Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project start + 2 months
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 12.8.3
11. CONTENT
The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, etc.) below.

1.1 Description/Purpose

The SDU Integration Plan provides a framework for the integration of SDU functionality into the CCSAS CSE system. This document will be used as a vehicle for the creation, documentation, and communication of the technical SDU integration solution, the activities for the SDU integration, roles and responsibilities, deliverables, audits and controls, and performance requirements.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions .

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable should be provided in this subsection.

2. Technical Solution Approach

This section shall describe the technical solution for supporting SDU integration.

3. Integration Approach

This section shall describe the functional and technical interaction between business entities and the business processes that span both the SDU vendor and the CCSAS application (e.g., customer service, payment identification, payment exception processing). This section will define the portions of shared processes belonging to each system and the approach to implementing these processes in a phased manner.

4. Responsibilities and Deliverables

This section shall define the Business Partner responsibilities and deliverables particular to SDU integration.

5. Audits and Controls

This section shall describe audits and controls for SDU integration.

6. Performance Requirements

This section shall identify the approach to defining and allocating performance requirements related to SDU and CCSAS application integration. In addition, this section will define the process of how the Business Partner will allocate these requirements between the SDU and CCSAS application.

1. DELIVERABLE IDENTIFIER TM 030-1

2. DELIVERABLE NAME CSE/SDU Interface Design Description

3. STANDARD J-STD-016-1995 G.2.2

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 7/29/2004 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 12.8.6

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

In addition to the content specified in the document standard, this document will contain the business process timeframe budget specifications for the SDU and CSE applications.

1. DELIVERABLE IDENTIFIER TM 030-2

2. DELIVERABLE NAME CSE/SDU Interface Design Description

3. STANDARD J-STD-016-1995 G.2.2

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 4/29/2005 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 12.8.6

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

In addition to the content specified in the document standard, this document will contain the business process timeframe budget specifications for the SDU and CSE applications.

1. DELIVERABLE IDENTIFIER TM 031-1

2. DELIVERABLE NAME External Entity Interface Design Description (EE IDD)

3. STANDARD J-STD-016-1995 G.2.2

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 7/29/2004 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 12.8.12

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

In addition to the content specified in the document standard, this document will contain the business process timeframe budget specifications for the SDU and CSE applications. It will also define the specifications governing how the SDU will transmit disbursement and Bank Activity data (e.g., check numbers, disbursement dates, bank clearing dates) to the CSE system.

1. DELIVERABLE IDENTIFIER TM 031-2

2. DELIVERABLE NAME External Entity Interface Design Description (EE IDD)

3. STANDARD J-STD-016-1995 G.2.2

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 4/29/2005 plus or minus 5 State Business Days (updated per CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 12.8.12

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

In addition to the content specified in the document standard, this document will contain the business process timeframe budget specifications for the SDU and CSE applications. It will also define the specifications governing how the SDU will transmit disbursement and Bank Activity data (e.g., check numbers, disbursement dates, bank clearing dates) to the CSE system.

1. DELIVERABLE IDENTIFIER TM 032
2. DELIVERABLE NAME Master Test Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project start + 4 months
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed upon by CCSAS Project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 12.9.1
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Master Test Plan provides an overview of testing efforts for the CCSAS CSE project. This document will be used as a framework for test activity preparation, resource allocation, development of performance criteria, and development of relative testing deliverable due dates.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions .

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Testing Concepts

This section shall identify broad testing concepts that apply to the CCSAS CSE project.

3. Testing Prerequisites

This section shall identify prerequisite activities and resources needed for testing.

4. Approach to Testing with External Entities

This section shall discuss the methods and approaches for coordinating with external entities.

5. Resource Constraints

This section shall identify resource constraints for the testing of the CCSAS CSE system.

6. Schedule Constraints

This section shall identify the schedule constraints for the testing of the CCSAS CSE system.

7. Types of Testing

This section shall describe the five types of tests, their entry and exit criteria, and their correlation to the CCSAS CSE system development lifecycle. The following tests shall be addressed:

- Unit tests
- Unit integration tests
- System tests
- Performance tests
- System qualification tests (to be completed by the State)

8. Testing Automation

This section shall identify the tools and methods of testing automation that will be utilized in the CCSAS CSE project.

1. DELIVERABLE IDENTIFIER TM 033
2. DELIVERABLE NAME Software Test Plan (STP)
3. STANDARD J-STD-016-1995 E.2.2
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION 8/13/2004 plus or minus 5 State Business Days (updated per CR-00050)
8. DATE OF SUBSEQUENT SUBMISSION 5/13/2005 plus or minus 5 State Business Days (updated per CR-00050)
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 12.9.2, TM 12.9.8, TM 12.9.9
11. CONTENT (updated per CR-00124a)

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner. The Business Partner shall update the STP with the revisions necessary for the Version 1 Schedule Adjustment that will document the modified testing approach.

1. DELIVERABLE IDENTIFIER TM 034-1 (CDL deleted per CR-2-00124a)
2. DELIVERABLE NAME
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

(CDL deleted per CR-00124a, dated 12/22/04)

1. DELIVERABLE IDENTIFIER TM 034-2

2. DELIVERABLE NAME System Verification Test Description

3. STANDARD J-STD-016-1995 H.2.1

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION First System Verification Test Readiness Review for Version 2 minus 1 month

8. DATE OF SUBSEQUENT SUBMISSION 30 State business days prior to each subsequent System Verification Test Readiness Review

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 12.9.6

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 035

2. DELIVERABLE NAME System Verification Test Report

3. STANDARD J-STD-016-1995 H.2.2

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 5 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 10 State business days after the completion of the first System Test Verification

8. DATE OF SUBSEQUENT SUBMISSION 10 State business days after the completion of each subsequent System Verification Test

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 12.9.7

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 036
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 037
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 038 (Total Revision of CDL per CR-00057)
2. DELIVERABLE NAME Pre-Implementation Readiness Assessment Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 5 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION One month prior to the first PIRA Submission cutover
8. DATE OF SUBSEQUENT SUBMISSION None, however at 3 Months, and 1 Month prior to implementation at each LCSA and at 2 Months and 1 Month for DCSS a Pre-Implementation Readiness Assessment (PIRA) Work Product will be submitted and one month after cutover at each LCSA and DCSS a Post-Implementation Review Report (PIRR) Work Product will be submitted
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 13.0.1, TM 13.0.2, and TM 13.1.7
11. PLAN CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The PIRA Plan describes the readiness assessment approach, methods, tools and monitoring. The BP will continue to present LCSA readiness at the Implementation Readiness Review (IRR) ten days prior to the LCSA's scheduled cutover subject to formal State review and acceptance. Information presented at the IRR will then be used by the State to make the final go/no-go decision for the LCSA cutover to CASES.

The PIRA Work Products submitted at 3 months and 1 month prior to cutover for each LCSA and at 2 months and 1 month for DCSS provide a mechanism to align expectations between CCSAS project staff, the Business Partner, and LCSA staff regarding the work to be done in the LCSA. The PIRA Work Product includes a checklist of Readiness Metrics, which are a set of criteria that drive the implementation preparation efforts in each LCSA. The completion of the PIRA Work Product is guided by the completion of the Tool Kit, a workbook that includes a series of templates organized by discipline. By completing the information in the Tool Kit, the site moves closer toward completing activities necessary for implementation.

For Phase I, a PIRA Work Product is completed for each of the 14 LCSAs that transition to CASES from KIDZ or STAR/KIDS and for DCSS, a state-level site. For Phase II, PIRA Work Products are completed for the 58 LCSAs and DCSS, including the transition of the California Central Registry (CCR) to the CCSAS CSE system. The PIRA Work Product is completed for the LCSAs and for CCR (State-level site in Phase II) at three months prior to cutover and then monitored at one month prior to cutover. The PIRA Work Product for DCSS (State-level site in

Phase I) is customized at 2 months prior to cutover and monitored at 1 month prior to cutover.

To provide context for the PIRA Work Product, review of prior implementations (PIRA Work Products) and (PIRR Work Products) will be considered.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Development of the PIRA Work Product

This section shall describe the overall method, approach and tools used to develop the PIRA Work Product. It shall identify the main activities to be conducted for the site and identify the timeframes associated with completing these activities in order to be ready by the cutover date. Additionally, it shall include Readiness Metrics and identify key site-specific issues and risks that will be addressed.

12. Pre-Implementation Readiness Assessment Template/Contents

1. Monitoring the PIRA Work Product

This section and the subsection and Appendices that follow shall describe the individual method(s) by which information is monitored and analyzed for the PIRA Work Product. It provides a framework to identify specific tasks to be completed and progress toward the completion of the activities. Site-specific tasks and other general system readiness areas will be monitored as part of the PIRA Work Product.

1.1 Work Backlog Assessment

This section shall describe how work backlogs are assessed and how the reviewed site performs in this area.

1.2 Data Conversion Assessment

This section shall describe how data the data conversion process is assessed and reported in the PIRA Work Product and how the reviewed site performs in this area. It shall include:

- Data cleansing assessment
- Summary of data clean-up activities
- Current conversion status
- Results of pre-cutover conversion testing

1.3 User Training Completion Review

This section shall describe how the completion of training is assessed and how the reviewed site performs in this area.

1.4 Local Interface Readiness

This section shall describe how local interface readiness is assessed by verifying connections and how the reviewed site performs in this area.

1.5 Help Desk Readiness

This section shall describe how help desk readiness is assessed and how the reviewed site performs in this area.

1.6 LCSA Personnel Technical Support Readiness

This section shall describe how technical support readiness of LCSA personnel is assessed and how the reviewed site performs in this area.

1.7 Hardware/Software Installation Site Preparation Review

This section shall describe how work site preparation reviews, to include hardware and software installation and network connectivity where applicable, are assessed and how the reviewed site performs in this area.

1.8 Completion of Old System Activities

This section shall describe how completion of old system activities is assessed and how the reviewed site performs in this area.

1.9 Work-In-Progress Assessment

This section shall describe how work-in-progress is assessed and how the reviewed site performs in this area.

1.10 Customer Service Readiness

This section shall describe how customer service readiness is assessed and how the reviewed site performs in this area.

2. Results of the PIRA

This section shall describe how outstanding pre-implementation issues, risks, and key post-assessment readiness activities that must take place prior to implementation will be handled.

This section shall also describe the next steps to be taken to address any issues identified. The PIRA Work Product submitted at 3 months shall contain the issues that pose the greatest risk to successful production operations and system use and assigns responsibilities for addressing these deficiencies. It additionally identifies key issues uncovered by the PIRA Work Product that will be added as tasks or issues to the transition strategy and/or project plan, revising the schedule and estimates as necessary.

Appendices

Appendix A – Planning Team Composition

This appendix shall provide a list of participants involved in completing the activities described in the Readiness Assessment for that particular site.

Appendix B – Risk Assessment Log

This appendix shall describe the site-specific risks and corresponding corrective actions identified and reported as a part of the PIRA Work Product. The Risk Assessment is created

with the initial delivery of the 3 month PIRA Work Product for a site and updated with the second subsequent delivery.

Appendix C – Issue Log

This appendix shall provide a log of implementation related issues. The issue list shall include for each open issue: an issue number, description of the issue, and the action item(s) and specific responsibilities assigned for the issue's resolution. The Issue Log is created with the initial 3 month delivery of the PIRA Work Product for a site and is updated with the second subsequent delivery to include newly identified issues and the status of issues documented in previous Work Products.

13. Post- Implementation Readiness Assessment Template/Contents

1.0 Implementation Review Results

This section and Appendix shall provide a review of the implementation results in the site, focusing on the following areas:

1.1 Issues and Problems Encountered During Site Implementation

This section shall describe the issues and problems that were encountered during the implementation process, as well as the reasons those problems occurred. This section shall also include a list of production issues that arose after implementation as a result of the transition, including any bridges that were implemented.

1.2 Timeframes Met and Missed

This section shall describe the planned and actual completion dates for each key implementation activity for the site. For those activities not completed on time (Slippage occurred), the reason(s) for slippage shall be identified.

1.3 Successes and Lessons Learned

This section shall describe the successes and any relevant lesson(s) learned.

2.0 Recommendation for Future Implementations

Based on the contents of Post IRA section 1, this section shall provide recommendations for future site implementations. For each problem or issue encountered where there is an opportunity for improvement, this section shall also provide a description of lessons learned.

Appendices

Appendix A – This appendix shall provide a data map for each converted system that maps current system date to the target system date.

1. DELIVERABLE IDENTIFIER TM 039 (CDL Deleted per CR-00057)
2. DELIVERABLE NAME
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

This CDL deleted with CR-00057, approved on February 26, 2004

1. DELIVERABLE IDENTIFIER TM 040
2. DELIVERABLE NAME Legacy Data Archive Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION 11/24/2003 (updated per CR-00017)
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 13.1.2
11. CONTENT
The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Legacy Data Archive Plan shall provide a description of the approach, methods, and strategies required to convert data from legacy systems to a statewide data archive.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions .

1.4. Referenced Documents

This subsection shall list the number, title, revision, date, and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Scope and Approach

This section shall describe the activities, approach, method, and tools for: a) the design and build of the legacy data archive database; b) the design and build of the application modules for access to the legacy data archive database functions and transfer of legacy archive data; c)

the design and build of legacy data archive automated conversion tools; d) the mapping and transfer of legacy archive data; e) the development of standard format descriptions; and f) the development and delivery of training materials for the use of the legacy archive database and the transfer and use of legacy archived data.

3. Statewide Summary of Legacy Systems

This section shall document the legacy data archive business rules as mutually agreed to by the State and Business Partner. The intent of the business rules is to identify which population of cases are candidates for the legacy data archive. The specific tables and elements to be populated shall be described in TM 041-1, Data Conversion Requirements Document.

1. DELIVERABLE IDENTIFIER TM 041-1
2. DELIVERABLE NAME Data Conversion Requirements Document
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION 11/14/2003 (updated per CR-00033 and CR-00050)
8. DATE OF SUBSEQUENT SUBMISSION 8/02/2004 (updated per CR-00033)
9. DELIVERY/FORMAT Electronic/In Accordance With CSE Project Tools
10. SOW PARAGRAPH REFERENCE TM 13.1.3, TM 13.1.4
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The purpose of the Data Conversion Requirements document is to define conversion requirements for the 58 counties, Non IV-D, POP, CCR, and IDB databases.

For Phase I, this document shall include data conversion requirements in the form of a target conversion data dictionary for the CASES system. This data dictionary shall include a description of the conversion tables, table elements, field formats, edits, validations, and valid value codes for the CASES system. It shall also include an assessment of rules for selection and conversion of data established by the State and Business Partner. It shall also include data conversion requirements established by the State and the Business Partner for Non IV-D data, POP data, and the legacy data archive.

For Phase II, this document shall include data conversion requirements in the form of a target conversion data dictionary for the new system. This data dictionary shall include a description of the conversion tables, table elements, field formats, edits, validations, and valid value codes for the new system. It shall also include an assessment of rules for selection and conversion of data established by the State and Business Partner. It shall also include data conversion requirements for CCR, POP, and IDB databases.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions .

1.4. Referenced Documents

This subsection shall list the number, title, revision, date, and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Conversion Requirements – CASES

This section shall document the conversion requirements for the CASES system in the form of a target conversion data dictionary. It shall also document an assessment of potential sources of conversion data, and location and volume of data.

3. Conversion Requirements – CCSAS CSE Application

This section shall document the conversion requirements for the CCSAS system in the form of a target conversion data dictionary. It shall also document an assessment of potential sources of conversion data, and location and volume of data.

4. Conversion Requirements – Other Data Sources

This section shall document the conversion requirements for the POP, CCR, IDB, and Non IV-D (NICE) databases.

5. Conversion Requirements – Data Archive

This section shall document the conversion requirements for the legacy data archive in the form of a target conversion data dictionary.

1. DELIVERABLE IDENTIFIER TM 041-2
2. DELIVERABLE NAME Data Conversion Requirements Document
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION 5/13/2005 plus or minus 5 State Business Days
8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan
9. DELIVERY/FORMAT Electronic/In Accordance With CSE Project Tools
10. SOW PARAGRAPH REFERENCE TM 13.1.3, TM 13.1.4
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The purpose of the Data Conversion Requirements document is to define conversion requirements for the 58 counties, Non IV-D, POP, CCR, and IDB databases.

For Phase I, this document shall include data conversion requirements in the form of a target conversion data dictionary for the CASES system. This data dictionary shall include a description of the conversion tables, table elements, field formats, edits, validations, and valid value codes for the CASES system. It shall also include an assessment of rules for selection and conversion of data established by the State and Business Partner. It shall also include data conversion requirements for Non IV-D data, POP data, and the legacy data archive.

For Phase II, this document shall include data conversion requirements in the form of a target conversion data dictionary for the new system. This data dictionary shall include a description of the conversion tables, table elements, field formats, edits, validations, and valid value codes for the new system. It shall also include an assessment of rules for selection and conversion of data established by the State and Business Partner. It shall also include data conversion requirements for CCR, POP, and IDB databases.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions .

1.4. Referenced Documents

This subsection shall list the number, title, revision, date, and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Conversion Requirements – CASES

This section shall document the conversion requirements for the CASES system in the form of a target conversion data dictionary. It shall also document an assessment of potential sources of conversion data, and location and volume of data.

3. Conversion Requirements – CCSAS CSE Application

This section shall document the conversion requirements for the CCSAS system in the form of a target conversion data dictionary. It shall also document an assessment of potential sources of conversion data, and location and volume of data.

4. Conversion Requirements – Other Data Sources

This section shall document the conversion requirements for the POP, CCR, IDB, and Non IV-D (NICE) databases.

1. DELIVERABLE IDENTIFIER TM 042-1

2. DELIVERABLE NAME Data Conversion Plan

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Project start + 5 months

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 13.1.4, TM 13.1.9, TM 13.1.11, TM 13.1.13, TM 13.1.17

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

This section shall state the intended use and audience of this document. It shall identify the systems or databases to be converted.

1.2 Document Overview

This subsection shall summarize the content of each major section of this Plan. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions .

1.4. Referenced Documents

This subsection shall list the number, title, revision, date, and source of the documents referenced within this Plan.

1.5 Related Plans, Processes and Procedures

This subsection shall identify the plans, processes and procedures impacted by, or that impact, this document and shall indicate the nature of the relationship to this document.

2. Roles and Responsibilities

This section shall identify each of the organizational entities involved in activities defined within this Plan. It shall indicate the involvement of each organizational entity in each activity by using a Responsibility, Approval, Consult, Inform (RACI) matrix.

3. Approach to Conversion Software Development

This section shall describe the approach, methods, tools, and strategies for developing the conversion software. It shall indicate whether the development of the conversion software is the modification of existing software or new development. This section shall:

a) Identify and describe the activities associated with conversion software development, to include the work products generated to plan, monitor, control, conduct, and verify those activities

b) Approach for defining conversion requirements to include the validation of existing conversion requirements, if appropriate

c) Approach to designing and development of the conversion software

d) Approach to testing the conversion software

4. Approach to Pre-Cutover Activities

This section shall describe the approach, methods, tools, and strategies for data conversion activities that occur prior to site cutover. This section shall:

a) Identify and describe the pre-cutover activities

b) Identify and describe the work products generated to plan, monitor, control, conduct, and verify pre-cutover activities, to include the frequency at which work products are generated, when they are produced and how they are used

c) Describe the approach, methods and tools used to map data between source and destination databases

d) Describe the approach, methods and tools used to perform data cleanup prior to cutover

e) Describe the approach, methods and tools used to plan, conduct and evaluate data conversion acceptance testing conducted prior to cutover to validate the results of the conversion process

f) Identify and describe the information provided to support the cutover readiness assessments

5. Approach to Post-Cutover Activities

This section shall describe the approach, methods, tools, and strategies for data conversion activities that occur following site cutover of a site. This section shall:

a) Identify and describe the post-cutover activities

b) Identify and describe the work products generated to plan, monitor, control, conduct, and verify post-cutover activities, to include the frequency at which work products are generated, when they are produced and how they are used

c) Describe the approach, methods and tools used to identify and resolve post-cutover exceptions

d) Describe the approach, methods and tools used to perform data cleanup following cutover

e) Identify and describe the information used to support the post-cutover evaluation of conversion activities

6. Approach to Converting Image Data

This section shall describe the approach, methods, tools and strategies used to convert image data.

1. DELIVERABLE IDENTIFIER TM 042-2

2. DELIVERABLE NAME Data Conversion Plan

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION First Version 2 Conversion minus 12 months

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 13.1.4, TM 13.1.9, TM 13.1.11, TM 13.1.13, and TM 13.1.17

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

This section shall state the intended use and audience of this document. It shall identify the systems or databases to be converted.

1.2 Document Overview

This subsection shall summarize the content of each major section of this Plan. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date, and source of the documents referenced within this Plan.

1.5 Related Plans, Processes and Procedures

This subsection shall identify the plans, processes and procedures impacted by, or that impact, this document and shall indicate the nature of the relationship to this document.

2. Roles and Responsibilities

This section shall identify each of the organizational entities involved in activities defined within this Plan. It shall indicate the involvement of each organizational entity in each activity by using a Responsibility, Approval, Consult, Inform (RACI) matrix.

3. Approach to Conversion Software Development

This section shall describe the approach, methods, tools, and strategies for developing the conversion software. It shall indicate whether the development of the conversion software is the modification of existing software or new development. This section shall:

a) Identify and describe the activities associated with conversion software development, to include the work products generated to plan, monitor, control, conduct, and verify those activities

b) Approach for defining conversion requirements to include the validation of existing conversion requirements, if appropriate

c) Approach to designing and development of the conversion software

d) Approach to testing the conversion software

4. Approach to Pre-Cutover Activities

This section shall describe the approach, methods, tools, and strategies for data conversion activities that occur prior to site cutover. This section shall:

a) Identify and describe the pre-cutover activities

b) Identify and describe the work products generated to plan, monitor, control, conduct, and verify pre-cutover activities, to include the frequency at which work products are generated, when they are produced and how they are used

c) Describe the approach, methods and tools used to map data between source and destination databases

d) Describe the approach, methods and tools used to perform data cleanup prior to cutover

e) Describe the approach, methods and tools used to plan, conduct and evaluate data conversion acceptance testing conducted prior to cutover to validate the results of the conversion process

f) Identify and describe the information provided to support the cutover readiness assessments

5. Approach to Post-Cutover Activities

This section shall describe the approach, methods, tools, and strategies for data conversion activities that occur following site cutover of a site. This section shall:

a) Identify and describe the post-cutover activities

b) Identify and describe the work products generated to plan, monitor, control, conduct, and verify post-cutover activities, to include the frequency at which work products are generated, when they are produced and how they are used

c) Describe the approach, methods and tools used to identify and resolve post-cutover exceptions

d) Describe the approach, methods and tools used to perform data cleanup following cutover

e) Identify and describe the information used to support the post-cutover evaluation of conversion activities

6. Approach to Converting Image Data

This section shall describe the approach, methods, tools and strategies used to convert image data.

1. DELIVERABLE IDENTIFIER TM 043
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 044
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 045
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 046
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 047
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 048
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 049
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 050
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 051
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 052
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 053 (CDL rewritten per CR-00064)
2. DELIVERABLE NAME Outreach Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project Start + 7 months and 2 weeks
8. DATE OF SUBSEQUENT SUBMISSION Project Start + 24 months; as necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 13.2.1
11. CONTENT

The first submission shall include two plans for LCSAs, one describing the minimum recommended outreach activities for successful CASES conversion, and the other describing the minimum recommended outreach activities for successful implementation of Statewide Services._The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

The Plans shall include a Resource Guide containing the content required in sections 1-9 below.

1. Introduction
This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.
1.1 Description/Purpose
The Resource Guide provides a document that describes the outreach activities to external entities (parties that are external to the CCSAS project, DCSS, and the LCSAs) that are affected by the CCSAS project and implementation of a new CSE system. The Resource Guide serves as the roadmap for development and delivery of messages to external stakeholders, including customers. It identifies and provides a timeline for the CCSAS outreach activities. This Resource Guide builds upon the information already assembled by DCSS and the CCSAS project staff regarding stakeholders and outreach.

The Resource Guide provides recommendations for outreach materials (including the method of production) that notify customers of changes in member and case information; provide addresses where payments are to be sent; define and provide contact information for questions and issue resolution; address customer access of the new CSE system; identify the dates and times that changes become effective; and define the specific outreach assessment activities and timing of messages. The Resource Guide will recommend change management strategies for customers and stakeholders, and will reflect recommended enhancements and modifications to improve existing communications based on best practices identified by the

Business Partner in communication and outreach efforts relative to other large-scale automation and business change projects.

1.2 Document Overview
This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.
1.3 Definitions
This subsection shall refer to a set of common project definitions.
1.4. Referenced Documents
This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.
1.5 Related Deliverables
This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. External Audience Analysis
This section shall identify and describe the targeted external audience groups and the information needs of each group based on input from the CCSAS CSE project and DCSS regarding child support stakeholders. This section shall describe the assessment methodology and activities conducted to analyze the audience groups' needs and the resulting information.

3. Outreach Methods
This section shall describe the outreach methods for delivering information to external audiences. This section shall describe for each specific audience the key messages, vehicle, sender, and timeframe for conducting outreach. The specific audience is the targeted group to receive the message. Key messages summarize the content of the information. Key messages are either informational, such as an introduction to the CCSAS CSE project, or instructional, such as how to access the new CCSAS eChild Support Customer Self-Service Application. The vehicle is the communication instrument used to deliver the message. The sender is the person(s) who delivers the information to the audience. The timeframe is the suggested frequency and schedule for the outreach activity.

4. Message Development Approach
This section shall describe the approach for developing and approving the specific messages for external audiences. This section shall include the roles and responsibilities of the CCSAS project staff, DCSS, and Business Partner team members.

5. Material Production Approach
This section shall describe the approach for producing outreach materials. Some outreach materials will be available from the DCSS web site. Other outreach materials will be produced for distribution from the CCSAS CSE project, DCSS or the LCSAs.

6. Statewide Outreach Delivery Approach
This section shall describe the approach for working with project sponsors to develop the statewide approach and materials and to deliver information statewide concerning the transition to a new CSE system.

7. Local Outreach Delivery Approach
This section shall include our approach to customizing a template of a local outreach plan for effective communication to local external stakeholders as we roll out the application to LCSAs. This section shall describe the approach for customizing a stakeholder analysis and outreach materials for each of the LCSAs that transition to a new CSE system. It also shall include the roles and responsibilities of the LCSA and combined CCSAS project/Business Partner staff in customizing and delivering local outreach.

8. Feedback Approach
This section shall describe the feedback and/or evaluation approach for monitoring statewide outreach activities. This includes how well outreach activities are being implemented, whether the correct messages are being communicated, and how audiences are reacting to the information, and how the plan will be updated based on feedback and/or evaluation information.

9. Outreach Timeline
This section shall describe the timeline for the outreach activities including audience analysis, development of messages, production of materials, statewide and local delivery, and feedback collection. The timeline for local outreach will be based upon the LCSA Rollout Schedule.

1. DELIVERABLE IDENTIFIER TM 054
2. DELIVERABLE NAME Orientation Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project Start + 7 months
8. DATE OF SUBSEQUENT SUBMISSION Project Start + 27 months; as necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 13.2.2, TM 13.2.3
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Orientation Plan is a user-oriented document that describes the activities to develop and deliver information to internal entities (personnel and staff internal to the State's child support enforcement program, such as DCSS and the LCSAs) that are affected by the CCSAS project and implementation of the new statewide CSE system. The Orientation Plan is used as the roadmap for development and delivery of information to these internal users and includes the approach for managing the impact of change on users.

The Orientation Plan is focused on DCSS and each of the existing county consortia. It includes planning for the orientation methodology, activities, roles, responsibilities, development of an orientation guide, use of the model office (in Phase II), and scheduled presentations.

The Orientation Plan describes the approach for delivering orientation to affected users. Phase I orientation activities will focus on the implementation of CASES with users in the KIDZ, and STAR/KIDS consortia counties and the implementation of Statewide Services with select DCSS users. It also will cover limited orientation communications with LCSA users from the 58 counties and DCSS regarding the Statewide Services component. Phase II orientation activities will focus on the implementation of the new CCSAS CSE system with users in the 58 LCSAs and with users in DCSS and the CCR.

1.2 Document Overview

This subsection shall summarize the content of each major section in the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents
This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.
1.5 Related Deliverables
This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Orientation Methodology
This section shall describe the methodology, activities, and roles and responsibilities for conducting orientation activities and using the Orientation Guide and the Model Office to conduct the scheduled orientation sessions. It shall also include content development for the Orientation Guide. The Orientation Guide includes the business process analyses.

3. Audience Analysis
This section shall identify and describe the internal audience groups and the information needs of each group. Internal audience groups include the different categories of personnel who will be using the new CSE system in LCSAs and state-level sites. This section shall describe the activities conducted to analyze the audience groups.

4. Orientation Methods
This section shall provide the orientation methods for delivering information to internal personnel. The orientation methods section shall describe the specific audience, key messages, vehicle, sender, and timeframe for orientation information. The specific audience is the targeted group to receive the message. Key messages summarize the content of the information. Key messages will be informational, such as an introduction to the CCSAS project, or instructional, such as how to access the CCSAS CSE Overview e-learning module that will be completed by CCSAS CSE users. The vehicle is the communication instrument used to deliver the message. The use of the As-Is and To-Be analyses and the Model Office to conduct orientation activities shall be described in this section. The sender is the person(s) delivering the information to the audience. The timeframe is the suggested frequency and schedule for delivering the message.

5. Message Development Approach
This section shall describe the approach for developing and approving the specific messages for internal audiences. This section shall include the roles and responsibilities of the CCSAS project staff, DCSS, LCSA, and Business Partner team members.

6. Material Production Approach
This section shall describe the approach for producing orientation materials. Some orientation materials will be available from the CCSAS project web site. Other orientation materials will be produced for distribution from the CCSAS project, DCSS, or the LCSAs.

7. Delivery Approach
This section shall describe the approach for customizing the orientation materials and information for each of the LCSAs that transition to CASES or the new statewide CSE system and for DCSS. Additionally, this section shall describe the approach for combined CCSAS project/Business Partner staff working with the LCSAs and DCSS to support the conduct of the orientation sessions.

7.1 Instructions
This section shall describe the instructions for using the Orientation Guide to perform orientation activities. It also shall include instructions for using the Model Office and conducting the user orientation session.

7.2 User Orientation Session Presenter Information
This section shall provide the instructional material for the user orientation session presenter. This section shall provide script, notes, bullet points, or tips on conducting the session and delivering the messages.

7.3 User Orientation Session Materials
This section shall contain the materials used during the user orientation session. The materials describe the project background and overview of the CCSAS system, functions, and features; introduce the new system functionality using the As-Is and To-Be models; explain the training methods; describe the transition activities; and describe the implementation timeline, site-specific business process changes, and implementation information, where applicable.

The materials include placeholders for site-specific business process changes or implementation information. Also, the orientation sessions shall build upon the customization of the baseline templates available in the Orientation Guide, including a summary of the key changes or impacts; description of the benefits of the new application and business process changes.

7.4 Other Orientation Templates
This section shall provide other orientation materials to communicate transition activities and business process changes to staff. Examples of other orientation materials templates include memos communicating the purpose, schedule, and logistics of a user orientation session; correspondence communicating progress of implementation activities; or meeting agendas for discussing changes to job activities with personnel.

8. Feedback Approach
This section shall describe the feedback and/or evaluation approach for monitoring orientation activities. This includes how well orientation activities are implemented, whether the correct messages are communicated, and how audiences are reacting to the information.

9. Orientation Timeline
This section shall describe the timeline for the specific orientation activities including audience analysis, development of orientation guide, material production, delivery, and feedback collection.

1. DELIVERABLE IDENTIFIER TM 055
2. DELIVERABLE NAME Orientation Guide
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project Start + 8 months
8. DATE OF SUBSEQUENT SUBMISSION Project Start + 28 months; as necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 13.2.2, TM 13.2.12
11. CONTENT
The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction
This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose
The Orientation Guide provides the materials for conducting user orientation sessions for personnel and staff internal to the State's child support enforcement program, such as DCSS and the LCSAs that are affected by the CCSAS project and implementation of CASES or the new statewide CSE system. The orientation sessions are used to describe the reasons for change, program improvements, local transition activities, training methods, and the timeline for implementation.
The focus of the Orientation Guide will be to document the differences in business processes under the existing system (the As-Is model) and the new system (the To-Be model). The Orientation Guide will provide baseline versions of the As-Is and To-Be models and templates that can be customized by each LCSA. These templates map the old business functions and terminology to the new business functions and terminology.
The Phase I Orientation Guide will include baseline system functionality maps that compare the KIDZ and STAR/KIDS systems (the As-Is models) to CASES (the To-Be model). The Phase II Orientation Guide includes baseline business process analyses comparing CASES and ARS (the As-Is models) to the CCSAS CSE system (the To-Be model), a gap analysis, and impact assessment. The Phase II Orientation Guide captures the CCSAS Business Model, which represents the future CSE Program uniform business processes to be implemented at the LCSA sties.

1.2 Document Overview
This subsection shall summarize the content of each major section in the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions
This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Orientation Guide Methodology

This section shall describe the methodology, activities, and roles and responsibilities for completing the Orientation Guide. This shall include the development of the business process analysis templates and the customization of the templates at the LCSAs.

3. Approach for Developing Business Process Baselines

This section shall describe the approach for developing and approving the business process baselines and templates. This section shall include the roles and responsibilities of the CCSAS project staff, DCSS, relevant Workgroups, LCSAs, and Business Partner team members.

4. Business Process Baselines

This section shall contain the baseline system functionality maps (Phase I) or the business process analyses (Phase II). For Phase I, this section shall include the baseline system functionality mapping of the two consortia systems being replaced by CASES. For Phase II, this section shall provide the baselines for As-Is business process flows (ARS and CASES), To-Be business process flows (under the new statewide CSE system), gap analysis, impact assessment, and documentation analysis.

5. Delivery Approach

This section shall describe the approach for using the orientation guide baselines to customize the As-Is and To-Be models for each of the LCSAs that transition to CASES or the new statewide CSE system and for DCSS. The customization effort will be guided by a set of templates. Additionally, this section shall describe the approach for combined CCSAS project/Business Partner staff working with the LCSAs to support the customization of the business process templates.

6. Business Process Analysis Timeline

This section shall describe the timeline for the specific business process analysis activities that occur in each site (LCSAs and DCSS) including developing and approving business process templates, developing the LCSA delivery approach, delivery, and feedback.

1. DELIVERABLE IDENTIFIER TM 056
2. DELIVERABLE NAME Business Process Transition Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project Start + 6 months
8. DATE OF SUBSEQUENT SUBMISSION Subsequent submission of BPTP for Version 2, Project Start + 24 months; All other subsequent submissions as necessary and mutually agreed-upon by CCSAS project staff and the Business Partner (updated per CR-00124a).
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 13.2.5, TM 13.2.7, TM 13.2.9, and TM 13.2.13
11. CONTENT
The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction
This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose
This Statewide plan shall describe the activities for transitioning business functions and shall provide a high-level transition plan for these business functions. It shall identify special considerations (if any) that need to be taken into account for specific consortia systems or sites (LCSAs and DCSS).

The Business Process Transition Plan provides a point of integration for the statewide transition of the LCSA CSE program operations and supporting DCSS and state-level business functions from the current environment to the new CCSAS CSE system environment. The plan integrates the transition of LCSA sites, the California Central Registry (CCR), Integrated DataBase (IDB), Statewide Disbursement Unit (SDU), and interfaces between the CCSAS system and external entities.

The Business Process Transition Plan also provides a point of integration for the CCSAS CSE project with critical milestones in the system development effort, DCSS business process improvement and re-engineering activities (e.g., P3 Project), and identified external influencers.

The Business Process Transition Plan will be developed for Phase I and reviewed and resubmitted as necessary for Phase II.

1.2 Document Overview
This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Scope and Structure of Business Process Transition Plan

This section shall introduce the Business Process Transition Plan and describe the components of the plan and how the components are integrated with other project activities and milestones. This section shall define the scope of activities addressed and coordinated with the Business Process Transition Plan.

3. Overview of CSE Present Business Environment

This section of the Business Process Transition Plan shall provide a high-level review of the present State of the California CSE business environment. This section shall identify the various business areas that provide CSE program services and entities with which the CSE program interacts, both internally and externally. These CSE services shall include existing LCSA consortia systems, and DCSS-delivered business processes and services provided through interagency agreements.

The purpose of this section shall be to establish and document the baseline “As-Is” CSE business environment in the State of California prior to the transition to the new CCSAS CSE system environment. The As-Is CSE business environment shall provide a system flow of business functions, as opposed to the business process flows that describe business processes performed in the system at the user level.

4. Overview of CSE Future Business Environment

This section shall provide an overview or the “big picture” of the CCSAS solution and its business functions. The CCSAS CSE solution provides a broad range of services and is used by, and interacts with, a large and varied audience. These services include the core child support program business functionality, an interface with the Statewide Disbursement Center (SDU), a Customer Service Center, a Help Desk, and interfaces with external agencies.

The purpose of this section shall be to document the “To-Be” end state business environment targeted by the CCSAS CSE project. This section of the plan shall establish the end of the system implementation phase of the project, where the business processes have been transitioned to the new CCSAS CSE environment. The To-Be CSE business environment shall provide a system flow, as opposed to the business process flows that describe business processes performed in the system at the user level.

5. Business Process Transition Approach

This section of the plan shall document the CCSAS CSE project’s approach to planning, piloting, executing, monitoring, safeguarding and evaluating the transition from present state to the new CCSAS CSE business environment and system. The plan shall identify the critical timeframes identified for critical business functions, such as the transition of the SDU, CCR,

IDB, and CPLS, and the impact of project, internal and external factors on the transition of those business functions.

This section of the plan shall also document the CCSAS CSE project processes for managing the transition in an integrated and coordinated manner. The purpose of these processes and procedures is to provide the mechanisms necessary to plan, schedule, execute, coordinate and evaluate the transition activities in high-quality, repeatable manner. The plan shall identify and document the parties and entities involved in the transition process and the roles and responsibilities of each of these parties and entities. The plan shall also document the processes and procedures that will be used to manage problem ownership and resolution during the transition process.

6. Managing the Transition and Potential Impacts
This section of the plan shall identify and describe the approach for managing processes that will be used to access and manage aspects of the transition such as: work backlogs, work-in-progress, potential operational disruption during transition, and close-out activities for existing system in use.

In addition to identifying the process that will be used to manage system-related aspects of the transition, this section of the plan shall also document the processes and procedures that will be used during the transition to manage and coordinate the “people” related impacts. This section shall identify how the approach defined in the Change Management section of the TMA will be coordinated with the overall transition effort to provide that the staff impact is reduced, the potential risks for operational disruption have been discussed with staff, and a plan for recovery and contingency is identified and included in the Business Rollback Plan.

The plan shall also identify the processes and procedures that will be used during the transition to manage and coordinate the use of “non-human” resources (e.g., hardware, supporting office equipment).

7. Business Rollback Plan
The Business Rollback plan is a component of the overall Business Process Transition Plan that shall provide the processes and procedures that shall be used in the event that a critical problem encountered during the transition requires a partial or full rollback of operations. This plan shall identify the key metrics and indicators that will be used to monitor the progress of the transition process from initial contact through the data conversion process, system implementation and transition of business processes to the new CCSAS CSE system. This section of the plan is coordinated with the technical aspects of rollback included in the Installation Contingency Plan (TM 071).

1. DELIVERABLE IDENTIFIER TM 057
2. DELIVERABLE NAME IDB Transition Analysis Report
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION 7/15/04, plus or minus 5 State Business Days
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner (updated per CR-00061)
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 13.2.6
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The purpose of this document is to record the analysis, findings, and recommendations regarding the transition of the Integrated Data Base (IDB) functionality to the new CCSAS CSE system. It provides a comparative analysis in which alternative solutions and corresponding business implications are evaluated. The IDB Transition Analysis Report will identify the constraints, dependencies, interactions, work processes, and data flows associated with the current IDB functions. This information will be used to develop a recommendation for integrating the functions of the IDB into the CCSAS CSE system. The results of this analysis are used in making a recommendation for the transition of the IDB, which is documented in this report.

The findings and recommendations presented in the report are used as input to the Business Process Transition Plan.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Analysis Methodology

This section shall describe the process of conducting the analysis. It shall describe the methodology used and the steps taken to conduct the analysis. The methodology will detail the approach to analysis the present business process environment and informational status of the IDB system and operation.

3. Analysis Results

This section shall describe the current (As-Is) state of the IDB processes and information. It shall include a description of the constraints, dependencies, interactions, work processes, and data flows associated with the IDB. The analysis will include particular focus on the documentation of “in-process” business flows and the impact on those flows of transitioning from the present operational environment to the CCSAS CSE system. The analysis will also identify and document the state of historical information maintained by the IDB to support on-going business processes.

4. Description of Alternative Transition Solutions

This section shall identify and describe the solution alternatives considered for the transition of IDB business processes and information to the CCSAS CSE system. It shall detail the elements of each alternative and associated business implications and shall provide information to explain how the alternatives were derived. The transition solution alternatives shall encompass the business process and work flows supported by IDB, and the information collected, processed, managed and stored by IDB.

5. Assessment of Alternative Solutions

This section shall assess and compare the alternative solutions for transitioning the IDB business processes and information to the CCSAS CSE system. It shall establish criteria for comparing the alternative solutions. This section shall document the evaluation of each alternative against the assessment criteria and provide a matrix summarizing how each of the alternatives examined fulfills the criteria. The product of the assessment is the identification of the recommended transition solution.

6. Transition Recommendation

This section describes the recommendation for transitioning the IDB to the CCSAS CSE system stemming from the IDB transition analysis. It shall identify the recommended solution and provide a detailed description of the solution and basis for the recommendation along with supporting information.

Appendices

Appendix A – Data Analysis Document(s)

This appendix shall provide the data analysis document(s) used in compiling and analyzing data for the assessment.

1. DELIVERABLE IDENTIFIER TM 058
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 059 (CDL Deleted per CR-00071)
2. DELIVERABLE NAME
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

This CDL deleted with CR-00071, approved on May 10, 2004

1. DELIVERABLE IDENTIFIER TM 060
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 061
2. DELIVERABLE NAME On-Site Implementation Support Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project Start + 7 months
8. DATE OF SUBSEQUENT SUBMISSION Project Start + 35 months; as necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 13.2.10
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The On-Site Implementation Support Plan describes the planning for the on-site support provided to LCSA users during implementation of CASES and the CCSAS CSE system. It provides an overview of the type, duration and level of support that each LCSA will encounter.

The On-Site Implementation Support Plan includes site-specific information on the number of resources needed to provide on-site support by classification and length of time, and by organization (CCSAS CSE project, the LCSA, or the Business Partner). The On-Site Support Plan also defines the roles and responsibilities that are necessary to deliver on-site support to the new system immediately following cutover and, where appropriate, on an ongoing basis as a part of the site's user support program.

The On-Site Support Plan is primarily focused on support provided to the LCSAs during the initial support period. As the models for on-site support differ between the two phases of the project, the plan is initially developed and submitted for Phase I and revised and submitted for Phase II. For Phase I, the plan includes support for the 14 LCSAs that transition to CASES from KIDZ or STAR/KIDS. For Phase II, the plan includes support for the 58 LCSAs that transition to the CCSAS CSE system. The plans for Phase I and for Phase II are submitted prior to beginning implementations in each phase. Additionally, the information specific to each site is reviewed and updated during the planning process for that site.

The plan will identify and describe on-site support activities and describe the approach, methods and tools used to plan, conduct and evaluate on-site support activities including activities related to pre-production verification to be conducted by the State.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key

components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Scope and Approach

This section shall describe the scope and general approach that will be used in the delivery of on-site support to the LCSA sites that transition to CASES or the new statewide CSE system. It shall include a description of the methods, tools, steps, and activities that will be used to manage, coordinate, and deliver the on-site support to the site during the transition period.

This section also shall describe the approach for on-site support, including the proposed level of support by phase and proposed schedule of on-site support. This section shall further describe the tasks and activities during the initial on-site support period and the approach for transitioning support responsibilities to the Help Desk and site superusers during the support period for the duration of the Business Partner's involvement in CCSAS CSE, called the ongoing support period. It shall describe the concept of "superusers," who are LCSA staff that are selected to help provide support in their own county. It also shall include issue resolution paths for initial and ongoing periods that differentiate between support that is provided on-site and the support provided by the Help Desk.

3. Statewide Roles and Responsibilities

This section shall describe the CCSAS statewide on-site support organization to include a definition of the roles and responsibilities of the key participants. The key participants, in this context, refer to groups or individuals responsible for major elements of on-site support on a statewide basis.

4. Site Roles and Responsibilities

This section shall describe the roles and responsibilities of the key participants in on-site support for each site. The key participants, in this context, refer to the groups or individuals that are assigned to a specific site for the purpose of providing on-site support services during post-cutover periods of the site's transition. This section shall further describe, where applicable, the roles and responsibilities for the ongoing support organization when established by the site.

5. Resource Allocation and Staff Plan

This section shall identify the number and type of users at each site needing support and the specific site-level participants in on-site support to include the identification and classification of support staff required (i.e. technical, subject experts); the number of staff required in each classification; the number of State employees required; the number of LCSA employees required; and the number of contractors required. It shall further define the timing associated with each classification and the longevity of support that each classification provides.

6. Staff Preparation and Development

This section shall describe the staff preparation methods and activities used to equip on-site

support staff with the tools and knowledge necessary to perform their job. Specifically, this section shall outline the process of orientation, training, and development that will be used to prepare the on-site support staff for the transition to CASES or the new statewide CSE system. This section shall also address the methods by which lessons learned during the transition period will be communicated and incorporated, if applicable, into the site's ongoing support program.

Appendices

Appendix A – Site-Specific On-Site Support Schedule

This appendix shall contain the site-specific, on-site support schedule by resource classification, and duration and by organization (CCSAS CSE project, LCSA, or Business Partner).

1. DELIVERABLE IDENTIFIER TM 062 – (combined with TM 064-2 with CR-2-00158-01)
2. DELIVERABLE NAME Train the Trainers Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Submittal of TM064-2 plus 20 State Business Days
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 13.3.2
11. CONTENT

Requirements for this CDL will be incorporated into TM 064-2

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Train-the-Trainer Plan contains the strategy for rolling out the Train-the-Trainer course. The Train-the-Trainer Plan consists of a serious of sections that contain the details of 'who, how, what, and when' trainers are identified and trained. Included are roles and responsibilities, training materials, timelines and milestones.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable should be provided in this subsection.

2. Roles and Responsibility

This section shall include the roles of the Business Partner Training Team, the CCSAS CSE project training team, and LCSA trainers. It shall also include the specific roles and

responsibilities of the individual team members.

3. Training Activities
This section shall include the detailed activities to Train-the-Trainers for the CCSAS CSE project and LCSA staff. Also included will be detailed activities addressing the formal training preparation session for Business Partner training staff. Included will be curriculum development, methodology, identification of trainees and scheduling of trainees for the Train-the-Trainer course.

4. Training Materials
This section shall include detailed information about the materials to be used during Train-the-Trainers. Included will be what materials will be developed, how materials will be developed and how trainers will use the materials. Included will be details about the Training Database, the Learning Management System (LMS) and Trainer Guides.
4.1 Training Database
This subsection shall include details about the training database and how it is to be utilized by trainers. Also included will be details on how the database is populated and how version control is coordinated with the system status.
4.2 Learning Management System (LMS)
This subsection shall include details about the Learning Management System (LMS) and how it is to be utilized by trainers. Also included will be details on how trainers can use the LMS for course registration, tracking and assessment of training delivered.
4.3 Trainer Guides
This subsection shall include details on how the Trainers Guides are developed and how they will act as a 'road map' for trainers as they deliver training.
4.4 Other Trainer Materials
This subsection shall include details on how the Trainers will use other developed materials, including the Software User Manual, Quick Reference Guide, On-line Help and the Frequently Asked Questions Guide.

5. Timeline and Milestones
This section shall include detailed information about the Train-the-Trainer timelines and milestones. Included will be details of when and where Train-the-Trainers courses will be offered and who should attend.
5.1 Timeline
This subsection shall include details about the timelines for the Train-the-Trainer course development and delivery.
5.2 Milestones
This subsection shall include details about the milestones for the Train-the-Trainer course development and delivery.
Appendices

Appendix A - Curriculum
This appendix shall include the curriculum goals, objectives and outline of the Train-the-Train course as described in section 3, Training Activities.
Appendix B - Timeline and Milestones
This appendix shall include the complete trainer timelines and milestones as described in section 5, Timeline and Milestones.

1. DELIVERABLE IDENTIFIER TM 063
2. DELIVERABLE NAME Training Transition Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Full System Implementation
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 13.3.3
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Training Transition Plan is a user-orientated document that describes the detailed activities for transitioning the training program infrastructure and curriculum to DCSS. The plan includes the details for the two-year post-implementation period and plans for final transition of training to the DCSS at the conclusion of the two-year post-implementation period. Also included in the plan are the transition roles and responsibilities, the transition of training materials, the timeline and milestones for transition, and a transition checklist.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Roles and Responsibility

This section shall include the roles of the Business Partner Training Team and DCSS during transition of training activities. It shall also include the specific roles and responsibilities of the individual team members.

3. Transition Activities

This section shall include the detailed activities to transition the training program infrastructure and curriculum to DCSS. Included shall be details for the two-year post-implementation period and for the completion of the transition. Also included shall be details on the incorporating lessons learned during the training.

4. Training Materials

This section shall include detailed information about the materials developed during training and the transition of those materials to DCSS.

4.1 Training Database

This subsection shall include details about the training database. Included shall be details on how the database is used for training, how it is populated, how version control is done to be in sync with the system status, and how it will be used during the two-year post-implementation period and during the final transition to DCSS.

4.2 Learning Management System (LMS)

This subsection shall include details about the Learning Management System (LMS) and how trainers utilize it for LMS course registration, tracking and assessment of training delivered. Also included shall be details on how the Learning Management System will be used during the two-year post-implementation period.

4.3 Other Training Materials

This subsection shall include details about the other training materials used during training delivery. Included shall be details about how to use and maintain the Software User Manual, Quick Reference Guide, On-line Help and the Frequently Asked Questions Guide.

5. Timeline and Milestones

This section shall include detailed information about the timelines and milestones for the 2-year post-implementation period and the final transition DCSS at the conclusion of the post-implementation period.

5.1 Timeline

This subsection shall include details about the timelines for the training transition. Included shall be timeline details for the two-year post-implementation period.

5.2 Milestones

This subsection shall include details about the milestones to be achieved for the training transition. Included shall be milestones for the two-year post-implementation period and for the completion of the transition.

6. Transition Completion

This section shall include detailed information about training activities that will for completion of the transition. Included shall be dates, activities, and responsibilities.

7. Transition Checklist

This section shall include detailed information about the training transition checklist. The

checklist will include detailed information to be completed for the two-year post-implementation period and for the completion of the transition

Appendices

Appendix A – Timeline and Milestones

This appendix shall include both the timeline and milestones for the two-year post-implementation period and for the completion of the transition as described in section 5, Timeline and Milestones.

Appendix B – Transition Checklist

This section shall include detailed information about the training transition checklist. Included shall be detailed checklists to be completed for the two-year post-implementation period and for the completion of the transition as described in Section 7, Transition Checklist.

1. DELIVERABLE IDENTIFIER TM 064-1

2. DELIVERABLE NAME User Training Plan

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION End of Consolidated Software Design Review (CSDR) plus 90 State calendar days (updated per CR-2-00119)

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 13.3.1, TM 13.3.4

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The CSE Version 1 User Training Plan is a user-oriented document that describes the details of the User Training effort for CSE Version 1 of the CCSAS project.

The CSE Version 1 User Training plan is submitted 9 months after project start. The CSE Version 1 User Training Plan is updated and re-produced as necessary as mutually agreed to by the State and the Business Partner.

1.2 Document Overview

This document is divided into sections as described in the following paragraphs:

Section 1: Introduction – This section presents identifying and background information about the document. It includes a description/purpose of the document, document overview, definitions, referenced documents, and related deliverables.

Section 2: Roles and Responsibilities – This section defines the roles and responsibilities for the Business Partner and the State Project User Training Teams.

Section 3: Training Methodology – This section describes the training methodology for developing and delivering CSE Version 1.

Section 4: Initial Training Curriculum – This section describes the approach for curriculum preparation, goals and objectives of curriculum and an outline of modules and courses that will be developed and delivered.

Section 5: Training Delivery – This section describes the training delivery approach of distributing information electronically.

Section 6: Training Materials – This section describes the training materials which consist of self-paced procedural information. This section describes how the CSE Version 1 training materials will be included in the CSE Version 2 on-line software user manual, if applicable.

Section 7: Training Schedule – This section describes the training schedule. It includes a list of the personnel types required to attend training. This section further identifies the timeline for distributing the training materials.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4 Referenced Documents

This section shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This section shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable should be provided in this section.

2. Roles and Responsibility

The section shall describe the roles of the Business Partner Training Team and the State project training teams.

3. Training Methodology

This section shall describe the Training Methodology to be used in developing and delivering CSE Version 1 training.

4. Initial Training Curriculum

This section shall describe the CSE Version 1 training curriculum that consists of Federal Reporting and Non IVD Orders. The goals and objectives and an outline of content will be included. The training will be developed in line with the functional architecture of the new CSE Version 1 application.

4.1 Federal Reporting

This subsection shall contain the goals, objectives and an outline of the Federal Reporting content.

4.2 Non IVD Orders

This subsection shall contain the goals, objectives and an outline of the Non IVD Orders content.

5. Training Delivery

This section shall include information about the distribution of the self-paced training materials electronically.

6. Training Materials

This section shall describe the self-paced procedural training materials.

7. Training Schedule

This section shall describe the training distribution schedule and timeline.

7.1 Personnel Required To Attend

This subsection shall include a list of personnel types required to attend training along with their titles and functions.

1. DELIVERABLE IDENTIFIER TM 064-2 – (includes TM 062 per CR-2-00158-01)
2. DELIVERABLE NAME User Training Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project Start + 22 months (updated per CR-00017)
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 13.3.1, TM 13.3.4, TM 13.3.10, and TM 14.0.6
11. CONTENT

This CDL includes requirements of TM 062 per CR-2-00158-01

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The User Training Plan is a user-oriented document that describes the details of the User Training effort for Phase II of the CCSAS CSE application. The User Training Plan consists of a series of sections that contain the details of 'who, how, where, what, and when' users are trained.

The User Training plan is a 'living document' that is periodically updated. As the training curriculum is developed, the User Training Plan will reflect updated information. As the training centers are secured, the User Training Plan will be updated to reflect Training Center information. The User Training plan is produced at 6-month intervals and delivered to the DCSS Training liaison for review and feedback.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Roles and Responsibility
The section shall include the roles of the Business Partner Training Team and the roles of the CCSAS CSE project training team. It shall also include the specific roles and responsibilities of the individual team members. Appendix A of this document shall contain the organization chart for training, which depicts the member level role and reporting structure.
2.1 Instructors
This subsection shall include the specific roles of the training team instructors. Included shall be the team member titles and skill levels.

3. Pre-Implementation Readiness Assessment
This section shall contain the particulars of the Pre-Implementation Readiness Assessment as it relates to training for each county. Included shall be the number of staff to be trained, computer skill level of the users, training course needs of the users, whether the county has existing training facilities and if the county has existing trainers or highly skilled workers to participate in the Train-the-Trainer program.

4. Training Methodology
This section shall contain the details of the Training Methodology to be used in Training CCSAS Phase II workers. Included shall be the specifics of our “two-fold” approach that includes both coach-assisted e-learning at the workplace and instructor-led e-learning at the training centers.

In addition, this section shall include Instructional Systems Design standards and concepts to help with the successful development and delivery of training.

5. Initial Training Curriculum
This section shall contain the approach for curriculum preparation, goals and objectives of curriculum and an outline of modules and courses that will be developed and delivered for the Initial CCSAS training of existing LCSA employees. Training curriculum modules will be developed in line with the functional architecture of the new CCSAS CSE application and with child support worker activity. As called out in the SCP, planning for the following subsections will be included in curriculum development.
5.1 CCSAS Overview
This subsection shall contain the goals, objectives and an outline of modules and courses for the CCSAS Overview curriculum. The CCSAS overview will include training on CCSAS CSE system navigation.
5.2 Case Management
This subsection shall contain the goals, objectives and an outline of modules and course for the CCSAS Case Management curriculum. Topics included in CCSAS Case Management will be case initiation, case update, case closure and scheduling.
5.3 Establishment
This subsection shall contain the goals, objectives and an outline of modules and courses for the establishment curriculum.
5.4 Enforcement
This subsection shall contain the goals, objectives and an outline of modules and courses for the enforcement curriculum.
5.5 Locate
This subsection shall contain the goals, objectives and an outline of modules and courses for

the locate curriculum.

5.6 Obligation Management

This subsection shall contain the goals, objectives and an outline of modules and courses for the Obligation Management curriculum.

5.7 Collections

This subsection shall contain the goals, objectives and an outline of modules and courses for the Collections curriculum.

5.8 Distribution

This subsection shall contain the goals, objectives and an outline of modules and courses for the Distribution curriculum.

5.9 Disbursement

This subsection shall contain the goals, objectives and an outline of modules and courses for the Disbursement curriculum.

5.10 Interstate

This subsection shall contain the goals, objectives and an outline of modules and courses for the Interstate curriculum.

5. 11 Archived Historical Data

This subsection shall contain the goals, objectives and an outline of the content for Archiving Historical Data, which will be incorporated into the appropriate modules.

5.12 Report Writing Tools

This subsection shall contain the goals, objectives and an outline of the content for Report Writing, which will be incorporated into the appropriate modules.

5.13 Customization

This subsection shall contain the goals, objectives and an outline of the content relating to customization, i.e. tables which will be incorporated into the appropriate modules.

5.14 System Use Policies

This subsection shall contain the goals, objectives and an outline of the content containing system use policies, i.e. file storage guidelines, equipment usage and software tool usage, which will be incorporated into the appropriate modules.

6. Follow-up Training

This section shall include detailed information about delivery for follow-up training of LCSA employees. Included will be details of how the Learning Management System (LMS) and the training database will be available for follow-up or refresher training once their original training is completed.

7. On-going Training

This section shall include detailed information about delivery and development of on-going or system update training. Included will be details of how the Learning Management System (LMS) and the training database will be updated and modified to reflect changes to the CCSAS CSE application. Also included will be details of ‘how’ and ‘what’ trainees will be notified of updated and changed training materials.

8. Training Delivery

This section shall include detailed information about the training delivery. Included will be which training modules receive coach led tutorials and which training modules receive classroom training. Training delivery approach will be discussed in detail, outlining scenario-based training, ‘Show Me-Let Me Try’ methodology, and module course development. Also included will be details on how the e-learning will be available for follow-up and ongoing

training past the initial training.

8.1 Classroom

This subsection shall include detailed information about the training delivery for those modules designated for classroom training. Included will be which trainees have been identified to receive classroom training, which modules will be delivered and how the scenario-based delivery approach will be used in the classroom.

8.2 Tutorial

This subsection shall include the tutorial or e-learning coach-assisted training delivery details. Included shall be which trainees have been identified to receive the e-learning training, which modules will be delivered and how the e-learning will be available for follow-up and ongoing training.

9. Training Materials

This section shall include detailed information about the materials to be used during training. Included will be what materials will be developed, how the materials will be developed and how materials will be distributed to trainees. Included shall be details for the trainer reference materials, on-line software user manual, quick reference guide, on-line help and frequently asked questions guide.

9.1 Training Scripts

This subsection shall include details on how the trainer scripts are developed and how they will act as a “road map” for trainers as they deliver training. Included will be the “hands-on” aspect of the trainer scripts, including processing all transactions on the training database.

9.2 Software User Manual

This subsection shall include the details of the Software User Manual (SUM). Included shall be details on how the SUM is developed, how the SUM contains a search engine, glossary, index and the use of hyperlinks.

9.3 Quick Reference Guide

This subsection shall include the details of the Quick Reference Guide. Included shall be details on how the Quick Reference Guide will be created, what information the guide will contain, and how the guide is disseminated to trainees.

9.4 On-line Help

This subsection shall contain the details of the On-line Help. Included shall be details of what tool is selected for on-line help, how the on-line help is developed and how it is used.

9.5 Frequently Asked Questions Guide

This subsection shall include the details of the Frequently Asked Questions (FAQ) Guide. Included shall be details on how FAQ are gathered and made available.

10. Training Tools

This section shall include detailed information about the tools to be used during training. Included in subsections shall be details about the “how, when and where” of the Learning Management System (LMS) and the Training Data Base.

10.1 Learning Management System

This subsection shall include detailed information about the Learning Management System (LMS) to be used in training. Included shall be the LMS product selection, features of the product and how those features will work with the training.

10.2 Training Lab

This subsection shall include detailed information about the training database. Included shall be details on how the training database is used, such as a training lab for integration with the LMS delivered scenario-based learning. Also included shall be details how the database will

be populated, how version control and changes are done to be in sync with the system status, and who is responsible for maintenance.

11. Training Development
This section shall include detailed information about the training development effort. Included is how work in progress is done and by whom. Also included shall be how the Business Partner Team follows the Instructional Systems Design methodology of analyze, design, develop, implement, evaluate and improve and how the designing of training materials, the software user manual, the e-learning modules and the training database are done in conjunction with the development of CCSAS.

12. New Worker Training Curriculum
This section shall include details for the New Worker Training curriculum development. Included shall be the goals, objectives and an outline for module and course development for the New Worker Training. Also included shall be details of "who" and "when" will receive new worker training.

13. Training Schedule
This section shall contain the details of the training schedule and scheduling process. In addition, this section shall include the current published training schedule, which details each training session to include the trainer, location, date, time, and staff required. This section shall further describe the timing and process used for schedule updates and the method(s) by which changes in the schedule are communicated to the State.
13.1 Personnel Required To Attend
This subsection shall include a list of personnel types required to attend training along with their titles and functions. This section shall further identify the specific training sessions each personnel type is required to attend to include the location, date, and time of each session.
13.2 Dissemination of schedule and classes
This subsection shall include details of how the training schedules and class details will be dissemination to trainees and supervisors via the CCSAS CSE project web site.

14. Training Center Facilities
This section shall contain the details associated with the establishment of the training center facilities. It shall include the location of each training center facility, the required furniture, fixtures, and training supplies. This section shall also include the details with respect to travel (directions from various locations) and local hotel accommodations for trainers and trainees.
14.1 Hardware
This subsection shall contain the details associated with the hardware establishment of the training facilities and centers. It shall include number of desktop PC's, printers, scanners, LCD projectors and placement of those items within the training facilities.

15. Training Evaluation
This section shall include details on the training assessment and effectiveness. It shall define and describe the methods by which trainees and trainers are evaluated and what is done with the results of the evaluations.

Appendices

Appendix A – Organizational Chart

This appendix shall include an Organizational Chart for the Business Partner Training Team and the Project Training Team as described in section 2, Roles and Responsibilities.

Appendix B – Curriculum

This appendix shall include the goals and objectives of curriculum and an outline of modules and courses as described in section 5, Initial Training Curriculum.

Appendix C – Training Schedule

This appendix shall include the complete training schedule. Included shall be county name, trainee names, trainee job functions, identified training course for each trainee, identified training center for each trainee, and identified dates for both classroom and e-learning for each trainee as described in section 13, Training Schedule.

1. DELIVERABLE IDENTIFIER TM 065
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 066
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 067
2. DELIVERABLE NAME Hardware and Software Installation Plan
3. STANDARD J-STD-016-1995 E.2.3
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project Start +12 months (updated per CR-00007)
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 13.4.1, and TM 13.4.8
11. CONTENT (updated per CR-00007)

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

The Hardware/Software Installation Plan (HISP) shall provide planning and installation information for each site where the Business Partner will install hardware and/or software.

The HSIP shall define and describe: (1) the planned hardware and software maintenance strategy and activities; and (2) the hardware and software refresh strategy and schedule.

1. DELIVERABLE IDENTIFIER TM 068

2. DELIVERABLE NAME Software Version Description (SVD)

3. STANDARD J-STD-016-1995 I.2.2

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 5 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION First System Qualification Test minus 10 State business days (updated per CR-00073)

8. DATE OF SUBSEQUENT SUBMISSION With each subsequent System Qualification Test and ORAR minus 10 State business days

9. DELIVERY/FORMAT Electronic/In Accordance With CSE Project Tools

10. SOW PARAGRAPH REFERENCE TM 13.4.2

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 069
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 070 – Converted to Work Product with CR-00084

2. DELIVERABLE NAME Release Notes

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE No formal State acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE N/A

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Draft Version of WP TM 070 (to include format of Release Notes only) due to State: 15 State business days prior to the first ORAR. Final Version of WP TM 070 due to State: 3 State business days prior to the first ORAR.

8. DATE OF SUBSEQUENT SUBMISSION Draft Version of WP TM 070 due to State: at the time when each Release goes to System Test. Final Version of WP TM 070 due to State: 1 State business day after System Qualification Test (SQT) or other acceptance test is complete for each Release, with the Release Notes being published when the Release goes into production.

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 13.4.4

11. CONTENT

CR00084 changes this CDL to a Work Product

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, the narrative shall be written in each of the three subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Release Notes document is used to communicate CCSAS CSE application changes introduced by a new release of the application software. Release Notes are published and distributed at the time the change is released into production. These notes communicate the specific details of the application change.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Heading Information
The heading information for the Release Notes shall identify the Release Number and the effective date and time of the change being described.

3. Change Description
The Change Description shall identify the application changes and the business impact of the change as identified by the Business Partner. This section shall also reference the entries in the Problem Management Tool and/or change requests included in the release.

4. Special Instructions
If there are any special instructions for the end user to perform as part of the release, such as work arounds, those instructions shall be detailed in this section.

1. DELIVERABLE IDENTIFIER TM 071 – (CDL deleted with CR-2-01016)
2. DELIVERABLE NAME
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 072
2. DELIVERABLE NAME Customer Service Support Center Assessment
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project start + 15 months
8. DATE OF SUBSEQUENT SUBMISSION N/A
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 14.0.2
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

The Customer Service Support Center Assessment will consider any early call center strategy piloted by DCSS and leverage the existing call center infrastructure. The Customer Service Support Center Assessment will consider use of existing LCSA equipment such as IVR. The Customer Service Support Center assessment will address access in multiple languages.

The Customer Service Support Center Assessment will include analysis of the potential customer service sites and approaches that considers regionalized call centers and also takes into consideration: economic factors (such as EZA, TACPA, LAMBRA, etc.), toll costs, bilingual employees, current LCSA call center employee base, and other socioeconomic factors that utilizes the optimal number of regional call centers.

The Customer Service Support Center Assessment will include overall (Rough Order of Magnitude) ROM costs for the design, development, implementation, maintenance and support of each alternative, including the State-selected alternative.

1. DELIVERABLE IDENTIFIER TM 073 (updated per CR-00233)
2. DELIVERABLE NAME Customer Service Support Center Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE N/A
5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project start + 25 months
8. DATE OF SUBSEQUENT SUBMISSION Not anticipated, but as necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 14.0.3, TM 14.0.4, TM 14.0.5, and TM 14.0.7
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

The plan will address the development of the State 1 Customer Service Support Center as described in the State's Business Case Analysis (BCA). The plan will deal with the following for the Customer Service Support Center:

- Provide essential functionality to ensure calls to statewide number are appropriately handled.
- Support Non IV-D customers
- Manage V1 – V2 transition period.

This plan will address development and implementation activities, roles and responsibilities, milestones, support center description, operations concept, organizational structure, staffing including start-up staffing, CSE System and program training, transition/knowledge transfer (as needed), and outreach.

The Business Partner's cost for the design, development, implementation, maintenance and support of the Customer Service Support Center will be refined from the Customer Service Support Center Assessment (CDL TM 072) and the subsequent Business Case Analysis (BCA). This refinement will detail those costs, on a line item basis as provided in the BCA, and will be within 25% of the submitted BCA ROM for the State selected alternative.

1. DELIVERABLE IDENTIFIER TM 074
2. DELIVERABLE NAME Customer Service Support Center Training Materials
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION First ORAR for Version 2 minus 6 months
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 14.0.6
11. CONTENT
The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

.

1. DELIVERABLE IDENTIFIER TM 075
2. DELIVERABLE NAME System Operation Manual
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION First ORAR minus 3 months
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 14.1.3, TM 14.1.22, TM 14.2.1, and TM 14.2.2
11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Systems Operation Manual describes the scheduled operations of the production system. It is the operator's instruction manual. It contains specific instructions on things an operator needs to do to manage the system on a daily basis. This includes the descriptions of daily, weekly, and other schedules, batch runs, process/job flows, error messages and their actions. It identifies the regular scheduled reports used to monitor and manage operation of the system. It also includes descriptions of scheduled administrative jobs, such as backups, to be run by operators or scheduled for automatic execution. Descriptions of administrative tasks provide instructions on how to run the job, and what to do in abnormal situations. The rationale behind administrative tasks and how they fit into the overall system architecture is contained in the TID and the SAP.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Software Summary

This section shall summarize the software environment from an operator's point of view. This section may reference other documents containing the relevant information. This section shall include:

- Description of the software application
- Identification of software files
- Description of software environment, as defined by J-STD-016-1995 J.2.3.3.3
- Description of the software organization and overview of operations, as defined by J-STD-016-1995 J.2.3.3.4.

3. Installation and Setup

This section shall describe procedures that the operator is required to perform to install the software on the equipment, to configure the software, to delete or overwrite existing files or data, and to enter parameters for software operation. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

4. Processing Procedures

This section shall identify and describe the processing procedures. It shall include:

- Backup and Restore Procedures, as defined by J-STD-016-1995 J.2.1.5.5
- Normal Stop/Start Procedures
- Emergency Stop / Start Procedures
- Recovery from errors and malfunctions, as defined by J-STD-016-1995 J.2.1.5.6
- Error messages and conditions, as defined by JSTD-016-1995 J.2.1.5.7

5. Operations

5.1 Operational Run Schedules

This section shall define the Daily, Weekly, Monthly, and other periodic scheduled activities

5.2 Operational Run Descriptions

This section shall describe the operational runs, and be structured as represented in J-STD-016-1995 J.2.3.5

5.3 Operational Contacts

This section shall identify points of contact and procedures to be followed to obtain assistance and report problems, as defined by J-STD-016-1995 J.2.3.3.7.

1. DELIVERABLE IDENTIFIER TM 076 (updated per CR-00083-2)
2. DELIVERABLE NAME Service Delivery Management Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION First ORAR minus 3 months
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 14.1.4, TM 14.1.5 and TM 14.1.21
11. CONTENT

The content of the Service Delivery Management Plan will now include reference to Retention Control Document (formerly CDL TM089) which was changed to a Work Product with CR00083-2. The Business Partner shall maintain a State approved list… of retention period for all application and operating system files and programs version (ACF H-5b) in a Retention Control Document and deliver the current copy to the State project records center.

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Service Delivery Management plan describes the ongoing process concerned with managing and running the day-to-day IT operations and implements the Performance and Capacity Management Plan (TM 086-1, TM 086-2). The Operations Management operational process is designed to manage the integration of processes, procedures, and technologies in order to deliver operational services in an effective and efficient manner. The Service Delivery Management Plan will be used by the combined CCSAS project team to understand production support and the underlying production support processes.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable should be provided in this subsection.

2. Account Management
This section shall describe the Account Management Process, which consists of a set of key business activities that focus on successfully delivering IT services, and fulfilling contracts in a manner that results in acceptable levels of customer satisfaction.

The account management process shall also describe the single point of contact for operational service delivery issues, processes, dependencies, problem escalation and key contacts. The process also describes the oversight of centralized services directly affecting users. These services include help desk, system management controls, backup and restore, problem management, system availability and system performance.

3. System Management Controls
This section shall describe the mission of the Systems Management Controls (SMC). Specifically, it describes how SMC is to support the production processes of the Business Partner. The Problem and Change Management Process is integrated into our business, to maximize system availability and reliability on behalf of our commercial and internal accounts.

The following system management controls processes are described, including the process scope, definitions, objectives and process flows (where appropriate):

- Problem Management is the process of detecting, reporting, analyzing, tracking, and correcting problems impacting managed I/T resources and/or its supporting environment
- Root Cause Analysis - The Root Cause is the factor, or factors, that when eliminated or changed, will prevent the reoccurrence
- Change Management - The process of planning, controlling, coordinating, executing and monitoring changes affecting the I/T production environment. Changes include any installation or alteration of hardware, licensed system software, non-licensed application software, procedures, and environmental facilities, which adds to, deletes from, or modifies the I/T production environment or its attached network.

4. Measurements and Reporting Process
The Measurements and Reporting operational process, which defines the process and identifies its scope, shall be described in this section. The Measurements and Reporting operational process is intended to provide a standardized approach identifying metrics, identifying the metrics capturing methods, and managing services with respect to Managing Data and Reports and Managing Measurements and the Reporting process. This section shall also identify metrics to be captured and how these metrics shall be used, including metrics related to system availability.

The method of negotiating service level and the adjustments to service levels shall be identified. The SMC process for service level management will be described.

The process shall identify the objectives, roles and responsibilities, process flow, measurements, and reporting narrative and process steps.

The following reports have been pre-identified:

- **4.1 Database Activity Report**
- **4.2 Service Level Attainment Report**
- **4.3 System Performance and Availability Validation Report**
- **4.4 System Utilization Report**
- **4.5 System Maintenance Report**

5. Operations Monitoring Process
This section shall describe the Operations Monitoring Process. Specifically, this section shall identify the objectives, roles and responsibilities, process flow, operations monitoring narrative and process steps. It shall address the interaction with Help Desk Processes necessary to measure system availability. It shall identify specific help desk information required and the method for obtaining that information.

Operations Monitoring is the process of delivering specific CCSAS services to meet Service Level Agreements. It defines the operational structure, automation, and controls; monitors production workload and exceptions; displays availability; and manages system resources based on workload and service commitments.

6. Situation Management Overview
This section shall present the Situation Management Overview. Specifically, this section shall identify the objectives, roles and responsibilities, process flow, network administration narrative and process steps.

Situation Management facilitates recovery from system/file problems and outages and standardizes the communications with customer concerning these problems and outages.

7. Roles and Responsibilities
This section shall identify the roles and responsibilities of the Service Delivery Management Plan and includes a responsibility matrix that defines service delivery activities and an indicator of either a “perform” or “assist” role for both the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 077
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 078
2. DELIVERABLE NAME System Security Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Project start + 6 months
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 14.1.6, TM 14.1.8, and TM 14.1.10
11. CONTENT

1. Introduction

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, the narrative shall be written in each of the three subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

IT Security as defined by the System Security Plan is the process of providing the security protection of physical, logical, and network assets associated with delivery of services. It involves the definition of security strategies, policies, and procedures; managing their implementation by other Standard processes; and monitoring their compliance.

The System Security Plan includes the process and procedure for the following IT security components:

- Develop and maintain security strategies, policies and procedures
- Communicate security strategies, policies, and procedures across the e-business Hosting Center
- Develop security implementation plans to meet strategies, policies, and procedures
- Manage security implementation performed by other processes, for example:

  Maintaining physical access to controlled areas
  Implementing server security
  Providing supervisor-level access to servers
  Granting access rights to server data
  Verifying and revalidating access
  Resetting user passwords
- Monitor compliance to security strategies, policies, and procedures by other processes
- Provide periodic reports as required

• Provide incident management for security breaches

This document provides the definition of security strategies, policies, and procedures. The System Security Plan describes the security process integration with other operational processes, security roles and responsibilities, IT Security narrative, and the IT Security process steps. The processes described in the System Security Plan continue until the conclusion of the contract.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions

1.4 Referenced Documents

This section shall list the number, title, revision, date, and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This section shall summarize related documents or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this section.

2. Accountability and Security Roles and Responsibilities

This section shall identify and define the specific roles involved in delivering or supporting the IT Security Operational Process and the responsibilities associated with each roles. The Security roles that shall be described in the section are:

• Physical Access Owner -- The Access Owner is responsible for the implementation and change of physical access controls.

• Logical Access Control Owner -- The Access Control Owner is responsible for the overall implementation of logical access controls.

• IT Security Manager -- The IT Security Manager is responsible for the maintenance of the Processes, Procedures, Policies, and Standards developed and employed for IT Security protection of CCSAS assets in the delivery of services.

• Network Security Manager -- The Network Security Manager role is responsible for implementing network security controls for data networks.

• Portable Media Custodian -- The Portable Media Custodian is responsible for the IT Security Controls associated with classifying, protecting, and tracking portable media.

• Security Monitor -- The Security Monitor role is responsible for on going monitoring and testing of operations to help confirm that IT Security practices are implemented and working.

3. Security Policy and Procedure Communication

The System Security Plan shall describe the integration of the IT Security process with other service delivery processes. There are two types of integration:

• Operational Processes that IT Security Calls - These integration items are the operational processes that provide process or procedures for IT security. This section shall include the

input from IT security and the output from the operational process.

• Operational Processes that Call IT Security - These integration items are the operational processes that IT security provides either process or procedures. This section shall include the input from other operational processes and the output from the IT security process.

4. IT Security Narrative

The IT Security Narrative, which describes the IT Security Operational processes in the System Security Plan, shall describe the objective, tools, prerequisites, inputs, outputs, the called-by operational processes and the operational processes called. Additionally, the IT Security Narrative shall describe the security education approach, incident impact, cause analysis and reporting fields.

5. Disaster Recovery and Business Continuity

This section shall describe the security considerations for disaster recovery and business continuity. For disaster recovery, the security considerations that will be defined are the protection of the business data when stored for disaster recovery purposes and security transit procedures in the event of a declared disaster.

This section shall describe the business continuity security processes, which includes the processes for recovery, retrieval and restoring of data that affect data and information security.

6. Data Classification

This section shall describe the levels of data classification and the owner responsible for setting the data classification level. For each classification level the security requirements for protecting that data will be specified.

7. Legal and Regulatory Requirements

This section shall describe the legal and regulatory requirements for the different levels of data either included in the CCSAS CSE system and those systems that communicate data to and from the CCSAS CSE system.

8. IT Security Process Steps

The IT Security Process Steps, a standard approach to security, shall describe the process for review and updates of security processes and procedures. These steps proceed in parallel to monitor security and manage incidents/violations and maintain the integrity of the security plan from the start of the delivery of services to contract termination. The IT Security Process Steps section shall provide a description of the specific process step including the tasks within each step and the dependencies between process steps. Steps that have sub-processes shall also be identified. This section shall define the following process steps:

• Monitor Security (Sub-process)
• Identify Incidents / Violations
• Manage Incidents / Violations (Sub-process)
• Update Security Plan
• Update Access Controls

- Maintain Security Plan (Sub-process)
- Implement Physical Access Controls (Sub-process)
- Implement Portable Media Controls (Sub-process)
- Implement Logical Access Controls (Sub-process)
- Implement Network Controls (Sub-process)
- Risk Assessment
- Perform Risk Assessment
- Risk Acceptance
- Communicate Security Bulletins as Required
- Complete Security Process Steps at contract termination

1. DELIVERABLE IDENTIFIER TM 079 (updated with CR-2-01072)
2. DELIVERABLE NAME System Security Assessment Report
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 5 State Business Days
6. FREQUENCY OF SUBMISSION ~~Quarterly~~ Semi-Annually
7. DATE OF FIRST SUBMISSION Full Implementation of Version 1 + 3 months
8. DATE OF SUBSEQUENT SUBMISSION N/A
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 14.1.8
11. CONTENT
The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The System Security Report describes any noted system vulnerabilities to fraud or theft, loss of data, physical destruction, unauthorized access, intrusion, and any captured harmful events or activities. This report includes the frequency of each risk analysis, including dates of both the last risk analysis and intended future risk analyses. The System Security Report will be provided on a ~~monthly~~ semi-annual basis.

The System Security Report will be reviewed by the Business Partner and CCSAS CSE project operations and management personnel to identify and close security gaps.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. System Security Metrics

This section shall describe the various system security metrics, including references to the

related process, procedures, or tools that will be used to monitor the system for security problems. This section will also contain the frequency of measurement or monitoring of the metrics. This section will also detail the frequency with which the system will be measured and monitored against the established System Security standards.

3. System Security Report Format

This section shall describe the System Security Report format and frequency of the report. The report will include the date, time, and identified source (if any) of security threats identified by the CCSAS CSE system. Where they have been identified, specific fraud, theft, loss or disclosure of data, unauthorized access or intrusion will be listed in this report along with the corrective action that was taken.

1. DELIVERABLE IDENTIFIER TM 080
2. DELIVERABLE NAME End User ID Procedures
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Full Implementation of Version 1 minus 1 month
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards
10. SOW PARAGRAPH REFERENCE TM 14.1.9
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Terminal User ID Procedures describes the key elements of the CCSAS CSE project's delegation and maintenance of the password authority and the notification process to quickly notify those responsible when personnel start, change within, or leave the project. These procedures will be used by the production support personnel, to maintain control over access to the CCSAS CSE system and the data that it contains.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Delegation and Maintenance of the Password System

This section shall describe the key elements of the delegation and maintenance of the CCSAS password system. The goal of this system is to limit the number of people with the ability to access the password system and to control the list of passwords. The Service Delivery Manager will own the process of assigning delegation and maintenance activities to only those Business Partner and CCSAS CSE project members that require this authority to perform their

job activities. The description will include the length of time between password changes of the password authority.

3. Personnel Change Notification Process
This section shall describe the key elements of the approach used to quickly notify the Service Delivery Manager, or the responsible party, when there are personnel changes on the project. These mechanisms can include checklists or other ways of identifying personnel rollover.

1. DELIVERABLE IDENTIFIER TM 081
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 082
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 083 (TM083 & TM084 combined per CR-2-00295)

2. DELIVERABLE NAME Local Desk Procedures for System Administration and Network Administration

3. STANDARD J-STD-016-1995 J.2.4

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION First Version 2 ORAR minus 3 months

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 14.1.12

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 084 (combined with TM083 per CR-2-00295)
2. DELIVERABLE NAME Local Desk Procedures for Network Administration
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 085

2. DELIVERABLE NAME Software User Manual Plan (SUMP) Name changed with CR-2-00182

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 8/22/05 date changed with CR-2-00182

8. DATE OF SUBSEQUENT SUBMISSION In accordance with the System Configuration Management Plan

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 14.1.14

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The CSE Version 2 Software User Manual (SUM) is a user-oriented electronic document that provides the user procedures for how to use the CCSAS CSE application to perform child support work activities. The SUM will be periodically updated to reflect system changes.

1.2 Deliverable Overview

This subsection shall summarize the content of the completed contract deliverable. This subsection shall describe that the SUM consists of online user procedures, glossary of terms, frequently asked questions, and problem reporting/support information. The subsection shall describe how the SUM information is integrated into the CCSAS CSE application.

The subsection shall describe the online user procedures are presented in context with the business processes. The online user procedures provide the user with the information and/or tasks to complete specific child support activities related or associated with the CSE CCSAS application window.

This subsection shall also describe the format of the procedures and the approach for linking or integrating the procedures with the CCSAS CSE application.

The subsection shall list the functional categories for which online procedure will be developed including case initiation and update, locate, establishment, enforcement, obligation management, collections, distribution, disbursement, central registry, federal reporting, ad hoc reporting, and system administration.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Access Instructions

This section shall provide the instructions of how to access the online user information and content within the CCSAS CSE application.

1. DELIVERABLE IDENTIFIER TM 086-1

2. DELIVERABLE NAME Performance and Capacity Management Plan

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Version 1 Operational Readiness Assessment Review (ORAR) minus 3 ½ months (updated per CR-00124a)

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 14.1.15, TM 14.1.6, TM 14.1.18

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

Capacity Planning, as described in the Performance and Capacity Management Plan, encompasses the process of planning for adequate IT resources required to fulfill current and future resource requirements. This process is limited to only providing capacity trends of operating system resources (CPU/Memory/Disk).

Performance Management, as defined in the Performance and Capacity Management Plan, is the discipline of measuring, analyzing, reporting and tuning the performance of operating system resources (CPU/Memory/Disk).

The table below defines the differences between the Performance Management process and the Capacity Planning process.

Performance Management	Capacity Planning
Improve performance of Resources	Improve use of resources
Reactive	Proactive
Short term	Long term
Logical	Physical
Response Time	Business Planning / Financials
Existing Resources	New Resources

The Performance and Capacity Management plan is developed prior to the production support activities taking place. The Performance and Capacity Management plan will be used by the combined CCSAS project team that is engaged in production support activities. This plan

describes the Capacity Planning and Performance Management activities and processes to be used to manage the eight CCSAS CSE environments.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Environments

The environment section shall list the CCSAS environments that the performance and capacity planning process will monitor. This section will describe the eight CCSAS CSE environments that will be monitored for performance and capacity management, with a focus on the production environment.

3. Performance and Capacity Plan Objectives

This section shall describe the performance and capacity plan objectives.

The objectives of this section are to describe the processes that provide the following services:

- Monitoring of system availability measures
- Analyzing performance measurements to determine problems
- Providing proactive analysis of system performance trends
- Providing tuning parameter recommendations
- Providing capacity trending and proactive analysis of system resources (CPU/Memory/Disk)

The definition of and process for creating the red server action list shall be included in the plan, and will include a process to maintain the appropriate the number of servers on the red server action list, which is automatically created via SRM.

4. Scope of the Performance and Capacity Planning Processes

This section shall describe the scope of the performance and capacity planning processes. The process includes the following activities:

- Process and procedures to monitor, measure, and report system performance and system capacity
- Identification of the Capacity Planning Lead and a description of their role in production support
- Define / Maintain Performance Alerts including the components and the metrics that will be monitored
- Provide IT Resources Balancing / Tuning / Workload recommendations to upgrade or replace hardware and software to meet contractual obligations
- Monitoring trends in a system performance and usage, to identify the need for system

changes due to user growth or state directed changes through the definition and analysis of Performance Reports on a monthly basis and more frequently during periods of rapid growth or change

- Provide analysis of system performance and usage and identify opportunities to improve system performance and capacity through changes in technology or business practice. Included is the collection of Performance statistics on a monthly basis
- Response to Alerts generated by the performance management toolset
- Manage the resolution of server operating system level performance problems through follow-up and escalation, with appropriate account support and/or support personnel
- Evaluation of storage management issues that affect performance management to maintain good performance
- System performance budget tracking
- The process for defining performance metrics will be identified
- Evaluate the efficiency of the CCSAS CSE load balancing approach

5. Dependencies

This section shall list the other processes that Performance and Capacity Planning is dependent upon.

6. Tools

The tools to be used for the project shall be defined in this section. These tools will monitor performance characteristics that are contractual obligations or assist in the monitoring and troubleshooting of the system. Elements to be monitored are: system availability, system disk space, CPU utilization and database metrics.

7. Process Measurements

This section shall describe the Process measurements which will define the performance metrics to be captured, the reporting frequency, and the tools used to capture and report each performance metric. It shall include the methodology for obtaining and collecting help desk information necessary to measure system availability. The metrics to be captured include the system performance budget measurements determined during development and performance testing.

The objective of process measurement is to identify and review all servers in the red condition. The ultimate goal is to have no servers in the red condition, other than those that have been agreed to as an acceptable operating condition (AOC).

8. Defects and Exposures

This section shall identify known process defects or exposures, if any, at the time that the plan is created.

Appendices

Appendix A -- Server Monitoring Thresholds

This Appendix shall include tables that describe the standard monitors that were setup by the Business Partner. The base monitors are divided into two groups, UNIX, and NT. Within each group, the monitors are further subdivided into specific monitoring areas: Applications, CPU, Disk, Memory, and processes. Each of these subgroups consists of one or more monitors.

1. DELIVERABLE IDENTIFIER TM 086-2 (updated per CR-2-00296)

2. DELIVERABLE NAME Performance and Capacity Management Plan

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 15 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION ~~First Version 2 ORAR (ORAR for Version 2 Pilot) minus one month~~ (updated per CR-2-00296) First Pilot County Cutover minus one month (updated per CR-2-01167)

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 14.1.15, TM 14.1.6, and TM 14.1.18

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

Capacity Planning, as described in the Performance and Capacity Management Plan, encompasses the process of planning for adequate IT resources required to fulfill current and future resource requirements. This process is limited to only providing capacity trends of operating system resources (CPU/Memory/Disk).

Performance Management, as defined in the Performance and Capacity Management Plan, is the discipline of measuring, analyzing, reporting and tuning the performance of operating system resources (CPU/Memory/Disk).

The table below defines the differences between the Performance Management process and the Capacity Planning process.

Performance Management	Capacity Planning
Improve performance of Resources	Improve use of resources
Reactive	Proactive
Short term	Long term
Logical	Physical
Response Time	Business Planning / Financials
Existing Resources	New Resources

The Performance and Capacity Management plan is developed prior to the production support activities taking place. The Performance and Capacity Management plan will be used by the combined CCSAS project team that is engaged in production support activities. This plan describes the Capacity Planning and Performance Management activities and processes to be used to manage the eight CCSAS CSE environments.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Environments

The environment section shall list the CCSAS environments that the performance and capacity planning process will monitor. This section will describe the eight CCSAS CSE environments that will be monitored for performance and capacity management, with a focus on the production environment.

3. Performance and Capacity Plan Objectives

This section shall describe the performance and capacity plan objectives.

The objectives of this section are to describe the processes that provide the following services:

- Monitoring of system availability measures
- Analyzing performance measurements to determine problems
- Providing proactive analysis of system performance trends
- Providing tuning parameter recommendations
- Providing capacity trending and proactive analysis of system resources (CPU/Memory/Disk)

The definition of and process for creating the red server action list shall be included in the plan, and will include a process to maintain the appropriate the number of servers on the red server action list, which is automatically created via SRM.

4. Scope of the Performance and Capacity Planning Processes

This section shall describe the scope of the performance and capacity planning processes. The process includes the following activities:

- Process and procedures to monitor, measure, and report system performance and system capacity
- Identification of the Capacity Planning Lead and a description of their role in production support
- Define / Maintain Performance Alerts including the components and the metrics that will be monitored

- Provide IT Resources Balancing / Tuning / Workload recommendations to upgrade or replace hardware and software to meet contractual obligations
- Monitoring trends in a system performance and usage, to identify the need for system changes due to user growth or state directed changes through the definition and analysis of Performance Reports on a monthly basis and more frequently during periods of rapid growth or change
- Provide analysis of system performance and usage and identify opportunities to improve system performance and capacity through changes in technology or business practice. Included is the collection of Performance statistics on a monthly basis
- Response to Alerts generated by the performance management toolset
- Manage the resolution of server operating system level performance problems through follow-up and escalation. with appropriate account support and/or support personnel
- Evaluation of storage management issues that affect performance management to maintain good performance
- System performance budget tracking
- The process for defining performance metrics will be identified
- Evaluate the efficiency of the CCSAS CSE load balancing approach

5. Dependencies

This section shall list the other processes that Performance and Capacity Planning is dependent upon.

6. Tools

The tools to be used for the project shall be defined in this section. These tools will monitor performance characteristics that are contractual obligations or assist in the monitoring and troubleshooting of the system. Elements to be monitored are: system availability, system disk space, CPU utilization and database metrics.

7. Process Measurements

This section shall describe the Process measurements which will define the performance metrics to be captured, the reporting frequency, and the tools used to capture and report each performance metric. It shall include the methodology for obtaining and collecting help desk information necessary to measure system availability. The metrics to be captured include the system performance budget measurements determined during development and performance testing.

The objective of process measurement is to identify and review all servers in the red condition. The ultimate goal is to have no servers in the red condition, other than those that have been agreed to as an acceptable operating condition (AOC).

8. Defects and Exposures

This section shall identify known process defects or exposures, if any, at the time that the plan is created.

Appendices

Appendix A -- Server Monitoring Thresholds

This Appendix shall include tables that describe the standard monitors that were setup by the Business Partner. The base monitors are divided into two groups, UNIX, and NT. Within each group, the monitors are further subdivided into specific monitoring areas: Applications, CPU,

Disk, Memory, and processes. Each of these subgroups consists of one or more monitors.

1. DELIVERABLE IDENTIFIER TM 087
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 088
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 089 – Converted to a Work Product with CR00083-2
2. DELIVERABLE NAME Retention Control Document
3. STANDARD
4. ACCEPTANCE No formal State acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE N/A
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Software Product Specification acceptance
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic/In Accordance With CSE Project Tools
10. SOW PARAGRAPH REFERENCE TM 14.1.21
11. CONTENT

CR00083-2 changes this CDL to a Work Product. Information required in this CDL will now be included in CDL TM076 – Service Delivery Management Plan.

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The Retention Control Document defines the amount of time that the CCSAS CSE components and data must be retained by the Contractor. This document is used by the operations staff to manage the archived and saved data.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Retention Periods

This section shall list retention periods for application and operating system files and program versions. The specific retention period, or minimum amount of time that this information is to be stored, shall be listed for the following CCSAS CSE project components:

- CSE program source code
- CSE database components
- CSE application data
- CSE application data containing IRS data
- CSE image data

As the owner of the CCSAS CSE application and data, it is the responsibility of the CCSAS project team to identify the retention period and provide that information to the contractor. The contractor will document the retention periods provided and place them in the Retention Control Document.

3. Retention Methods

This section shall define the retention methods, which may include tape, optical disk, microfiche, and paper

1. DELIVERABLE IDENTIFIER TM 090
2. DELIVERABLE NAME Disaster Recovery Plan
3. STANDARD Business Partner format with State acceptance
4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Version 1 ORAR minus 3 months
8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 14.1.27, and TM 14.1.28
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the deliverable. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The disaster recovery process, described in the Disaster Recovery Plan, is the process involved with the minimization of impact of, and recovery from unplanned interruptions of critical business functions. These interruptions, by their nature or anticipated duration, require a special level of recovery services beyond those that are required to resolve more typical outages.

The scope of disaster recovery encompasses the protection, recovery planning, and testing of plans to recover assets and facilities to support critical business functions including, but not limited to, servers, minicomputers, mainframes, data networks, data, applications, skilled personnel and their required production environments.

In the event of a disaster, the Business Partner shall provide disaster recovery services in accordance with this Disaster Recovery Plan and assume responsibility for operating the equipment and providing services at the primary recovery facility.

The Disaster Recovery Plan will be used by the Business Partner operations personnel as a guide during the twice-yearly disaster recovery testing. It will also be used in the event of a declared disaster to identify the systems to be recovered, the steps to be performed during that recovery and the roles and responsibilities of the Business Partner and the State.

The Disaster Recovery Plan will provide the plans and procedures for the restoration of the CCSAS CSE production environment and the network to connect the primary recovery center to the CCSAS CSE end users. No disaster recovery services are provided for the location CCSAS infrastructure in the 58 counties.

1.2 Document Overview
This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.
1.3 Definitions
This subsection shall provide the definition of those terms used in the completed deliverable whose meaning must be derived in context or that may be specific to the CCSAS project effort, LCSAs, or the Business Partner. The following definitions may be included in this section as appropriate:
• Disaster Recovery - The process of performing CCSAS CSE operations at an alternate site if the primary site is unusable because of a catastrophic event
• Primary Recovery Center - The IBM Business Recovery Site that is designated to perform CCSAS CSE operations in the event of a declared disaster
• Responsibility Matrix - Activities and responsibilities for Disaster Recovery Services
• BRS - Business Recovery Services is the organization within IBM Global Services that provides the primary disaster recovery center and the disaster recovery services
1.4. Referenced Documents
This subsection shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.
1.5 Related Deliverables
This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Disaster Recovery Processes
The Disaster Declaration Processes section shall identify the key personnel from the combined CCSAS project team with the authority to declare a disaster, and the contact information (Name, Phone and pager). The Disaster Declaration Process shall describe the responsibility of the Business Partner when a disaster is declared, including the establishing of a disaster command center and the communication methods between the affected parties.

The Disaster Recovery Plan shall include the list of steps to be performed during the transition of services from the primary site to the recovery site. The Disaster Recovery Plan shall include the steps for the move and restore of the production environment as soon as possible back to the primary data center after such facilities have been repaired, restored or replaced or to a mutually agreed upon alternate site.

The Disaster Recovery assumptions shall be listed. These assumptions shall include:
• Timeframe for access to the primary recovery center including a minimum and maximum number of hours after disaster declaration until the primary recovery center is available to start the recovery process. The requirement for disaster recovery is to have the production environment available 48 hours after a declared disaster
• Identification of the Primary recovery center
• Assumptions that describe the scope of the services to be covered under this Disaster Recovery Plan

The Disaster recovery processes shall identify the following:
• The backup facility - The current backup facility in case of a declared disaster is Boulder, CO.

This facility is geographically dispersed from the primary site to reduce the likelihood that an outage would affect both facilities
• Disaster recovery procedures for restoration of central CCSAS CSE operations for the production environment - These detailed steps will describe the system preparation, operating system installation, CCSAS CSE application code installation, data recovery, any system configuration activities required and the testing process
• Security - Any security issues will be identified
• Documentation of approved backup arrangements - Since the backup location is a Business Partner-owned and operated facility, the documentation of the disaster recovery arrangements is the formal agreement of all parties to provide services from the IBM Business Recovery Services organization
• An established processing priority method - The identification of a priority scheme within the CCSAS CSE application set
• Equipment and Software - A list of equipment and software to be used in the event of a disaster and for twice-yearly tests.

3. Critical Services
The critical services section shall list and describe the applications and application components that are covered under this Disaster Recovery Plan.

4. Disaster Recovery Roles and Responsibilities
The disaster recovery roles and responsibilities section shall include a responsibility matrix that defines disaster recovery activities and an indicator of either perform or assist role for both the combined CCSAS project team. The major categories of activities in the responsibilities matrix are: Disaster Recovery Planning, Disaster Recovery Plan Management, Disaster Recovery Test, Disaster Operations and Disaster Recovery Fees. Disaster Recovery Fees are charges that are incurred for the use of the disaster recovery center during its hosting of CCSAS CSE operations.

The Disaster Recovery Plan shall include a contact list of the combined CCSAS project team on the recovery management team who will be responsible for coordinating recovery-planning activities. Support teams shall be identified, and the names of individuals who are authorized by each party to declare a disaster defined in Section 2.

5. Disaster Recovery Testing Process
The Disaster Recovery Testing Process shall identify that disaster recovery testing will occur twice a year, the number of hours of concurrent annual test time, and the disaster recovery test procedures that describe the determination of the scope of each individual disaster recovery test. In addition, this section will identify the process for display lessons learned during the testing process.

Appendices

Appendix A -- Hardware and Software Configurations
The Hardware and Software Configurations appendix shall define the systems to be used, or the equivalent systems, in the event of a declared disaster. The detail of the hardware configurations shall include the customer name, CPU, primary recovery site, system memory, DASD Gigabytes and the quantity/Type Model/Description for options installed on the system. Software required shall be listed.

Appendix B -- Policies and Procedures required by the State of California Department of Information Technology
This appendix shall include the disaster recovery policies and procedures identified by the State.
Appendix C -- Policies for Operational Recovery Planning - October 2000
This appendix shall include the disaster recovery policies for operational recovery planning identified by the State.
Appendix D -- Procedures for the development of Operational Recovery Plans - October 2000
This appendix shall include the disaster recovery procedures for the development of operational plans identified by the State.

1. DELIVERABLE IDENTIFIER TM 091
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 092
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 093

2. DELIVERABLE NAME Software Transition Plan (STrP)

3. STANDARD J-STD-016-1995 E.2.4

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Full system implementation

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 14.2.3

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 094

2. DELIVERABLE NAME Help Desk Plan

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Help Desk Roll-Out minus 3 months

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 14.3.1, and TM 14.3.7

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The CCSAS Help Desk will resolve CCSAS end users incoming calls by answering these help desk calls, authenticating users, capturing call information in a Trouble Ticket system, assigning a problem severity, and resolving and tracking the call as defined in this Help Desk Plan.

The CCSAS CSE Help Desk will include three levels of support. Trouble Tickets are resolved on each level and are passed to the next level for resolution if the problem requires more knowledge. Each level of the Help Desk has more focused skills and the number of problems that require attention by each level of the Help Desk reduces from level 1 to level 2 and onto level 3. The target for first call resolution is 70%.

The Help Desk Plan will be developed as a key to performing the knowledge management component of CCSASCSE and to build the skills within the CCSAS project staff to operate the Help Desk upon transition.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4. Referenced Documents

This subsection shall list the number, title, revision, date, and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This subsection shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this subsection.

2. Call and Incident Processes

This section shall describe the Business Partner Help Desk process. This shall identify the closed-loop help desk process to be implemented using a process flow diagram and process narrative. The section shall show the call and incident processing components of the process flow and will identify any deviations from the standard processes that will be implemented for CCSAS CSE. The process will identify the following:

- Call Process and Distribution - The process for a call that is received at the Help Desk and routed to an available agent.
- Incident Identification and Classification - The process for gaining information on the problem and classifying it into one of four levels. Severity 1 problems have the highest priority and indicate a major malfunction of all system components without a workaround. Severity 2 problems have a major impact to selected system components. Severity 3 problems have a minor impact to system components and Severity 4 problems are problems that are considered enhancements.
- Incident assignment - After an attempt to resolve the problem, the level 1 help desk agent will assign the problem to a resolver group with more detailed knowledge.
- Incident escalation - The process for problems that have aged to have their severity increased to increase visibility and reduce the time to resolution.
- Incident Tracking - Communication meetings by CCSAS CSE Help Desk management to review open trouble tickets and trends in problem resolution.

3. Help Desk Integration with Level 2/Level 3 and local Help Desks

The Help Desk integration component section shall identify the level 2 and level 3 resolver groups for the Help Desk Services, describe the interface between the level 1 and the resolver groups and it shall identify a contact for each resolver group for the escalation of problems concerning the delivery of service.

The approach for the interfacing with local Help Desks, as part of CCSAS production support, shall be identified.

The interaction between the CCSAS CSE and local Help Desks will be defined by establishing a contact point for each County Help Desk. The exact roles and responsibilities for the CCSAS Help Desk and any supporting local help desk shall be identified.

4. Service Level Management and Reporting

The Service Level Management and Reporting section shall describe the Help Desk tools used for service level reporting and the processes for managing resources during the delivery of service. Service Level Management includes the process for the Help Desk to define and meet the established CCSAS CSE service levels, monitor and report compliance with these service levels.

The reports generated for the help desk shall include average call duration, call drop percentage, average call wait time, call volume and the percent of calls resolved upon first call. Help desk reporting will identify current and historical metrics. Actual help desk reporting fields

will be negotiated and can be modified upon mutual agreement of the Business Partner and the CCSAS project.

5. Customer Communication and Customer Satisfaction Measurements

The Customer Communications and Customer Satisfaction Measurements section shall include a brief description of the methods for communicating to the CCSAS end users that will have access to the CCSAS CSE Help Desk and a description of the customer satisfaction measurement process and methods.

6. Trouble Ticket System Users

This section shall describe the types of users of the Trouble Ticket system, such as the help desk agents and State management personnel. It shall describe the type of access each user shall be provided according to their authority.

7. Help Desk Planning

The Help Desk Planning section shall include the high-level project plan for the setup of the Help Desk including any key dependencies on the CCSAS project team. The scope of the Help Desk training activities shall be described along with an initial staffing plan for the Help Desk Services.

Help Desk Planning shall include the approach to knowledge management and providing the skills to the CCSAS project team to facilitate the transfer of help desk services. This will be provided by maintaining access to the CCSAS CSE Help Desk staff and by locating the CCSAS CSE Help Desk in the Sacramento California area during the transition period. The Business Partner Help Desk staff will lead help desk education classes and the help desk agent training will be provided.

8. Web Site for FAQs

This section will include a description of the relationship between the Help Desk and the Web Site for FAQs.

1. DELIVERABLE IDENTIFIER TM 095
2. DELIVERABLE NAME Deleted
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

1. DELIVERABLE IDENTIFIER TM 096–Converted to Work Product with CR00083-3
2. DELIVERABLE NAME Technical Review Agenda
3. STANDARD Business Partner Format with State Approval
4. ACCEPTANCE No formal State Acceptance
5. TIMEFRAME FOR STATE ACCEPTANCE N/A
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Technical Review minus 15 State Business Days
8. DATE OF SUBSEQUENT SUBMISSION N/A
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 8.3
11. CONTENT

CR00083-3 changes this CDL to a Work Product. Information required for this CDL will continue to be included in Technical Progress Review (TPR).

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

1.1 Description/Purpose

1.2 Document Overview

1.3 Definitions

1.4. Referenced Documents

1.5 Related Deliverables

2. Meeting Logistics

2.1 Meeting Name

2.2 Meeting Date(s)

2.3 Meeting Time(s)

2.4 Meeting Location

2.5 Invited Participants

3. Meeting Purpose

4. Agenda Items

5. Open Action Items

6. Open Issues

7. General Information

1. DELIVERABLE IDENTIFIER TM 097
2. DELIVERABLE NAME Technical Review Minutes
3. STANDARD Business Partner Format with State Approval
4. ACCEPTANCE State review and acceptance required
5. TIMEFRAME FOR STATE ACCEPTANCE 5 State business days
6. FREQUENCY OF SUBMISSION N/A
7. DATE OF FIRST SUBMISSION Technical Review + 5 State business days
8. DATE OF SUBSEQUENT SUBMISSION N/A
9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation
10. SOW PARAGRAPH REFERENCE TM 8.3
11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction
- 1.1 Description/Purpose
- 1.2 Document Overview
- 1.3 Definitions
- 1.4. Referenced Documents
- 1.5 Related Deliverables

2. Meeting Logistics
- 2.1 Meeting Name
- 2.2 Meeting Date(s)
- 2.3 Meeting Time(s)
- 2.4 Meeting Location
- 2.5 Attendees
- 2.6 Distribution List
- 2.7 Minutes Author
- 2.8 Minutes Preparation Date

3. Meeting Purpose
4. Meeting Agenda
5. Topics Discussed
6. Issues
- 6.1 New Issues
- 6.2 Open Issues

7. Decisions Made
8. Action Items
- 8.1 New Action Items
- 8.2 Open Action Items
- 8.3 Closed Action Items

9. General Information

1. DELIVERABLE IDENTIFIER TM 098

2. DELIVERABLE NAME CASES Training Plan

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE Walk-thru prior to delivery - State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Project Start + 9 months

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed-upon by CCSAS project staff and the Business Partner

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 13.3.1

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. Introduction

This section shall be divided into the following subsections to present identifying and background information on the standard. There shall be no written narrative under this heading. Rather, narrative shall be written in each of the subsections (1.1, 1.2, 1.3, 1.4, and 1.5) below.

1.1 Description/Purpose

The CASES Training Plan is a high-level document that describes the details of the User Training effort for Phase I. The CASES Training Plan consists of a series of sections that contain the 'who, how, where, what, and when' users are trained.

1.2 Document Overview

This subsection shall summarize the content of each major section of the completed contract deliverable. The summary for each section shall include a brief description of the key components of the section.

1.3 Definitions

This subsection shall refer to a set of common project definitions.

1.4 Referenced Documents

This section shall list the number, title, revision, date and source of the documents referenced in the completed contract deliverable.

1.5 Related Deliverables

This section shall summarize related deliverables or deliverables impacted by this deliverable. The deliverable identifier, SOW paragraph reference, deliverable name, and deliverable description for each related deliverable shall be provided in this section.

2. Roles and Responsibility

The section shall contain the roles of the Business Partner Training Team, the roles of the CCSAS CSE project training team, the roles of the CASES Consortium lead county and the LCSAs. It shall also include the specific roles and responsibilities of the individual team members.

3. Pre-Implementation Readiness Assessment

This section shall contain an overview of the Pre-Implementation Readiness Assessment as it relates to training for each county.

4. Training Curriculum

This section shall contain an overview of the approach for curriculum preparation, goals and objectives of curriculum and an outline of courses.

5. Training Delivery

This section shall contain general information about the training delivery. Included will be a general overview of which training will be delivered, to whom it will be delivered, and when it will be delivered. Also included will be the CASES Consortium lead county Train-the-Trainer approach to training.

6. Training Materials

This section shall contain general information about the materials to be used during training. Included will be what materials are available, and how materials will be distributed to trainees.

7. Training Tools

This section shall contain general information about the tools to be used during training. Included will be an overview of the training databases or training labs.

8. Training Schedule

This section shall contain general information about the training schedule and scheduling process. In addition, this section shall include the current published training schedule, which details when each county will be trained.

9. Training Center Facilities

This section shall contain an overview associated with the establishment of the training center facilities.

10. Training Evaluation

This section shall contain an overview of the training assessment and effectiveness. It shall define and describe the methods by which trainees and trainers are evaluated and what is done with the results of the evaluations.

11. Training Transition

This section shall contain an overview of the training transition from CASES training effort to the new best of breed CSE system.

Appendices

Appendix A – Organizational Chart

This appendix shall include an Organizational Chart for the Business Partner Training Team and the CCSAS CSE project team, the CASES Consortium lead county and the LCSAs as described in section 2, Roles and Responsibilities.

Appendix B – Training Schedule

This appendix shall include the complete training schedule. Included shall be county name, and dates when the particular county will be trained as described in section 8, Training Schedule.

1. DELIVERABLE IDENTIFIER TM 099-1 - (CDL Converted to Work Product with CR-2-00124a)

2. DELIVERABLE NAME System Performance Test Description

3. STANDARD J-STD-016-1995 H.2.1

4. ACCEPTANCE No Formal State Acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE N/A

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION Version 1 Operational Readiness Assessment and Review (ORAR) minus 4 ½ months

8. DATE OF SUBSEQUENT SUBMISSION 30 State business days prior to each subsequent Version 1 System Verification Test Readiness Review

9. DELIVERY/FORMAT Electronic/In Accordance with CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 12.9.12

11. CONTENT

CR-2-00124A changes this CDL to a Work Product

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 099-2

2. DELIVERABLE NAME System Performance Test Description

3. STANDARD J-STD-016-1995 H.2.1

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 20 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION First System Verification Test Readiness Review for Version 2 minus one month

8. DATE OF SUBSEQUENT SUBMISSION 30 State business days prior to each subsequent Version 2 System Verification Test Readiness Review

9. DELIVERY/FORMAT Electronic/In Accordance with CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 12.9.12

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 100

2. DELIVERABLE NAME Legacy Database Archive Design Description

3. STANDARD J-STD-016-1995 G.2.3

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 10/15/2004 plus or minus 5 State Business Days (updated per CR-00042 and CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION As necessary and mutually agreed upon by CCSAS project Staff and the Business Partner

9. DELIVERY/FORMAT Electronic/In Accordance With CCSAS Office Automation Standards

10. SOW PARAGRAPH REFERENCE TM 13.1.2

11. CONTENT

The content and format of the standard may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

1. DELIVERABLE IDENTIFIER TM 101

2. DELIVERABLE NAME ARS/CASES Functional Requirements Design Document (CDL name changed per CR-2-00108-01)

3. STANDARD Business Partner format with State acceptance

4. ACCEPTANCE State review and acceptance required

5. TIMEFRAME FOR STATE ACCEPTANCE 10 State Business Days

6. FREQUENCY OF SUBMISSION N/A

7. DATE OF FIRST SUBMISSION 7/29/04 plus or minus 5 State Business Days (updated per CR-00023 and CR-00050)

8. DATE OF SUBSEQUENT SUBMISSION N/A

9. DELIVERY/FORMAT Electronic & Hard Copy/In Accordance With CCSAS Office Automation

10. SOW PARAGRAPH REFERENCE TM 0.9

11. CONTENT

The content and format below may be tailored as mutually agreed upon by the CCSAS project staff and the Business Partner.

The initial submission will not contain the specifications for interfaces with Statewide Services, as this is dependent on specifications documented in the Version 1 External Entity Interface Design Description. The subsequent submission of this deliverable will contain the specifications for interfaces with Statewide Services.

This document shall:

1._The document shall specify requirements and design for ARS and CASES modifications necessary to meet federal certification requirements as documented in the CAP/GAP review, change requests, and RQs, and necessary to integrate with Version 1 Statewide Services and the SDU (updated per CR-2-00108-01).

2. Specify design to satisfy requirements

3. Document traceability between requirements and design

1. DELIVERABLE IDENTIFIER TM 102 Deleted with CR-2-00108-01
2. DELIVERABLE NAME
3. STANDARD
4. ACCEPTANCE
5. TIMEFRAME FOR STATE ACCEPTANCE
6. FREQUENCY OF SUBMISSION
7. DATE OF FIRST SUBMISSION
8. DATE OF SUBSEQUENT SUBMISSION
9. DELIVERY/FORMAT
10. SOW PARAGRAPH REFERENCE
11. CONTENT

This CDL deleted with CR-2-00108-01, approved on July 12, 2004