System Design Document (SDD)

Summary Description:

The System Design Document (SDD) describes how the functional and nonfunctional requirements recorded in the Requirements Document, the preliminary user-oriented functional design recorded in the Concept of Operations (ConOps) , and the preliminary data design documented in the Logical Data Model are transformed into more technical system design specifications from which the system will be built. The SDD is used to document both high-level system design and low-level detailed design specifications. The SDD describes design goals and considerations, provides a high-level overview of the system architecture, and describes the data design associated with the system, as well as the human-machine interface and operational scenarios. The high-level system design is further decomposed into low-level detailed design specifications for each of the system's components, including hardware, internal communications, software, system integrity controls, and external interfaces.

Status:

Mandatory – All new system development projects and any automated systems undergoing new major architectural design or functional changes (including GOTS and/or COTS integrations) must prepare some form of a System Design Document (SDD), as appropriate. The actual content of the SDD will be dependent on the specific circumstances of the IT project.

Timeframe:

The System Design Document (SDD) may be incrementally and iteratively produced during various stages of the system development life cycle, based on the particular circumstances of the IT project and the system development methodology being employed for developing the automated system. If an automated system is developed in multiple builds or releases, a SDD will likely be required for each major release. The SDD is generally initiated and baselined during the Design & Engineering Phase and serves as primary input to the Preliminary Design Review (PDR) and the Detailed Design Review (DDR). The SDD should be placed under configuration control and subsequently updated during the Development Phase and Implementation & Testing Phase, as necessary and appropriate. Information provided in the SDD serves as a basis for development of the Code, Implementation Plan, Test Plan, Test Case Specification (if applicable), Training Plan (if applicable), User Manual (if applicable), and Operator Manual.

Responsible Reviewing Component:

OIS/PMSG is the CMS component that has the primary decision authority over the need for the System Design Document (SDD), requirements for its creation, and acceptance of the end product in meeting the information needs.

Primary Information Exchange Partners:

The following are the primary stakeholders who have an interest in the content of the System Design Document (SDD):

Project Owner/Manager
System Owner/Manager
System Developer
OIS/ITAPS
OIS/ISMG
OIS/EDG
OIS/PMSG
OIS/SSG
OIS/TMG
IT Infrastructure Implementation Agent or Contractor
IV&V Contractor

Government Responsibilities:

The Project Owner/Manager is responsible for working with their designated Component Lead and OIS/PMSG to determine the level of detail required in the System Design Document (SDD) for the IT project or automated system. If a system is being developed in-house without outside contractor resources, then the government developers are responsible for the contractor responsibilities described below.

The Project Owner/Manager is responsible for planning, coordinating the activities, and facilitating the review of the system design that is developed in-house. Key government stakeholders are responsible for reviewing the content of the SDD and any supporting documents, based on their individual areas of subject matter expertise, and identifying any issues or concerns. The Project Owner/Manager or the Component Lead should record and prioritize the comments received from the reviewers in a Comment Form (Excel). All critical issues should be captured as Priority 1 comments. All recorded comments should be sorted by priority, page, and paragraph. The completed SDD Comment Form should be shared with all of the primary stakeholders.

Contractor Responsibilities:

The contractor serving as System Developer prepares the System Design Document (SDD) based on the functional and nonfunctional requirements documented in the Requirements Document, the preliminary user-oriented functional design recorded in the Concept of Operations (ConOps) document, if one exists, and the preliminary data design documented in the Logical Data Model, if one exists. The System Developer should perform an informal review of the SDD at the contractor's site prior to delivery of the SDD to CMS for formal review. The System Developer should also use the content of the SDD as the basis for subsequent system development activities, and to familiarize new development team members with the problem domain and the system to which the SDD applies.

If the system is being developed in multiple builds or releases, the entire system design may not be fully defined until the final build or release. The System Developer's planning should identify the portion of the system design that is to be defined in each build or release, and should prepare a formal Release Plan for the IT project. System design for a given build or release should be interpreted to mean defining the portion of the system design identified for that build or release.

The System Developer shall participate in defining and recording system-wide design decisions (i.e., decisions about the system's behavioral design and other decisions affecting the selection and design of system components). Design decisions remain at the discretion of the System Developer unless formally converted to requirements. The System Developer is responsible for fulfilling all formal requirements and demonstrating requirements traceability to the individual design components. Design decisions act as internal System Developer "requirements" to be implemented, imposed on subcontractors (if applicable), and confirmed by System Developer internal testing, but their fulfillment need not be demonstrated to CMS.

Content:

The System Design Document (SDD) needs to be as detailed as possible, while at the same time not imposing too much of a burden on the System Developer that it becomes overly difficult to create or maintain. The following represents the basic outline of a standard SDD. The ordering of the sections in the SDD attempts to correspond to the order in which issues are typically addressed and in which decisions are made during the design process. Though it is understood that system designs don't always proceed in this order (or in any linear, or even predictable order), it is useful to present them as if they did for the purpose of comprehending the design of the system.

Title Page
Revision Chart
Table of Contents
List of Figures
List of Tables

1. **Introduction**
2. **Referenced Documents**
3. **System Overview**
4. **Design Considerations**
   - **4.1. Assumptions and Dependencies**
   - **4.2. General Constraints**
   - **4.3. Goals and Guidelines**
   - **4.4. Development Methods and Contingencies**
   - **4.5. Architectural Strategies**
5. **System Architecture**
   - **5.1. Hardware Architecture**
   - **5.2. Internal Communications Architecture**
   - **5.3. Software Architecture**
   - **5.4. System Architecture Diagram**
6. **Data Design**
   - **6.1. Data Objects and Resultant Data Structures**

6.2. File and Database Structures
6.2.1. Database Management System Files
6.2.2. Non-Database Management System Files
7. Human-Machine Interface
7.1. Inputs
7.2. Outputs
8. Operational Scenarios
9. Detailed Design
9.1. Hardware Detailed Design
9.2. Internal Communications Detailed Design
9.3. Software Detailed Design
10. System Integrity Controls
11. External Interfaces
11.1. Interface Architecture
11.2. Interface Detailed Design
Appendices
a) Software Architecture Diagrams
b) Data Dictionary
c) Requirements Traceability Matrix
d) Software Product 508 Compliancy Checklist
Glossary

See System Design Document (SDD) Content (PDF - KB) for a detailed description of the expected content for each of the major components of the SDD. Some sections of the SDD (e.g., Data Design and External Interfaces) may be better addressed in separate, subordinate documents, such as a Database Design Document and an Interface Control Document (ICD). Note also that a Requirements Traceability Matrix and a Software Product 508 Compliancy Checklist (Word Document) are to be included as appendices to the SDD.

Guidance:

The Office of Information Services (OIS) is developing a set of IT technical standards and guidelines to assist System Developers in ensuring consistency and interoperability among the software applications, components, and services that comprise the CMS Infrastructure. While designing your automated system or application, please consider the CMS IT-Related Standards that are available and applicable to your project. If you need additional assistance and guidance in designing your system or application, contact your designated Component Lead, who will put you in touch with a representative from OIS to assist you.

Review Process:

The team that was involved in drafting the System Design Document (SDD) should conduct an informal review prior to the performance of any formal walkthrough(s) and final acceptance. The purpose of this incremental review process is to identify and resolve any defects or issues prior to baselining in an effort to achieve final agreement and acceptance of the system design by the stakeholders. A Comment Form should be used to record comments received from the review(s).

The information contained in the SDD is utilized by the Primary Information Exchange Partners listed above at various times throughout the IT investment and system life cycles. The high-level system design serves as primary input to the Preliminary Design Review (PDR). The low-level detailed design serves as input to the Detailed Design Review (DDR).

Date Created/Modified:

February 2005/May 2005

System Design Document (SDD) Content

The following is an outline and description for each of the major components of the System Design Document (SDD):

Title Page
Revision Chart
Table of Contents
List of Figures
List of Tables

1. Introduction

[Provide identifying information for the existing and/or proposed automated system or situation for which the SDD applies (e.g., the full names and acronyms for the development project, the existing system or situation, and the proposed system or situation, as applicable). Summarize the purpose of the document, the scope of activities that resulted in its development, its relationship to other relevant documents (e.g., the Requirements Document, Concept of Operations (ConOps), Logical Data Model, Interface Control Document (ICD), Database Design Document, and Release Plan, if they exist), the intended audience for the document, and expected evolution of the document. Also describe any security or privacy considerations associated with use of the SDD.]

2. Referenced Documents

[Provide identifying information for all documents used to arrive at the design and/or referenced within the SDD (e.g., related and/or companion documents, prerequisite documents, relevant technical documentation, etc.).]

3. System Overview

[Briefly introduce the system context and the basic design approach or organization, and discuss the background to the project. Provide a brief overview of the system and software architectures and the design goals. Explain what the proposed system will do (and not do, if necessary). Describe the relevant benefits, objectives and goals as precisely as possible. Include the high-level context diagram(s) for the system and subsystems previously provided in the Concept of Operations (ConOps) and/or Requirements Document, updated as necessary to reflect any changes that have been made based on more current information or understanding. If the high-level context diagram has been updated, identify the changes that were made and why.]

4. Design Considerations

[Describe many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.]

4.1. Assumptions and Dependencies

[Describe any assumptions or dependencies regarding the system, software and its use. These may concern such issues as: related software or hardware, operating systems, end-user characteristics, and possible and/or probable changes in functionality.]

4.2. General Constraints

[Describe any global limitations or constraints that have a significant impact on the design of the system's hardware, software and/or communications, and describe the associated impact. Such constraints may be imposed by any of the following (the list is not exhaustive):

a) Hardware or software environment
b) End-user environment
c) Availability or volatility of resources
d) Standards compliance
e) Interoperability requirements
f) Interface/protocol requirements
g) Licensing requirements
h) Data repository and distribution requirements
i) Security requirements (or other such regulations)
j) Memory or other capacity limitations
k) Performance requirements
l) Network communications
m) Verification and validation requirements (testing)
n) Other means of addressing quality goals
o) Other requirements described in the Requirements Document]

4.3. Goals and Guidelines

[Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system and its software. Examples of such goals might be: an emphasis on speed versus memory use; or working, looking, or "feeling" like an existing product. Guidelines include coding guidelines and conventions. For each such goal or guideline, describe the reason for its desirability unless it is implicitly obvious. Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system (e.g., choice of which specific product to use).]

4.4. Development Methods and Contingencies

[Briefly describe the method or approach used for the system and software design (e.g., structured, object-oriented, prototyping, J2EE, UML, XML, etc.). If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of these methods. If several methods were seriously considered, then each such method should be mentioned, along with a brief explanation of why all or part of it was used or not used. Describe any contingencies that might arise in the design of the system and software that may change the development direction. Possibilities include lack of interface agreements with outside agencies or unstable architectures at the time the SDD is prepared. Address any possible workarounds or alternative plans.]

4.5. Architectural Strategies

[Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe

the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Describe compliance with CMS Enterprise Architecture and standards. Specifically identify any deviations that were made from the CMS Enterprise Architecture and standards, and provide rationale to support the deviation(s). When describing a design decision, discuss any other significant alternatives that were considered, and the reasons for rejecting them (as well as the reasons for accepting the alternative finally chosen). Sometimes it may be most effective to employ the "pattern format" for describing a strategy. Examples of design decisions might concern (but are not limited to) things like the following:

a) Use of a particular type of product (programming language, database, library, commercial off-the-shelf (COTS) product, etc.)
b) Reuse of existing software components to implement various parts/features of the system
c) Future plans for extending or enhancing the software
d) User interface paradigms (or system input and output models)
e) Hardware and/or software interface paradigms
f) Error detection and recovery
g) Memory management policies
h) External databases and/or data storage management and persistence
i) Distributed data or control over a network
j) Generalized approaches to control
k) Concurrency and synchronization
l) Communication mechanisms
m) Management of other resources]

5. System Architecture

[Provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves in this section. A subsequent section of the SDD will provide the detailed component descriptions. The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality. At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portion of the system) must play. Describe how the system was broken down into its components/subsystems (identifying each top-level component/subsystem and the roles/responsibilities assigned to it). Describe how the higher-level components collaborate with each other in order to achieve the required results. Provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Make use of design patterns whenever possible, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them. Provide rationale for choosing a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality.]

5.1. Hardware Architecture

[Describe the overall system hardware and organization, indicating whether the processing system is distributed or centralized. Identify the type, number and location of all hardware components (e.g., servers, workstations, processors, backup

systems, output devices, etc.), with a brief description of each item and diagrams showing the connectivity between the components. Include resource estimates for processor capacity, memory, on-line storage, and auxiliary storage.]

5.2. Internal Communications Architecture

[Provide a detailed description of the system's communications network, denoting the communications architecture(s) being implemented (e.g., Token Ring, Ethernet, etc.) and how system components are linked. Describe any Local or Wide Area Networks and buses. Include descriptions of necessary equipment, such as hubs, routers, cabling, transmitters, firewalls, ports, etc.). Provide a diagram depicting the communications path(s) between the system and subsystem components. Include resource estimates for communications network capacity (LAN and WAN) needed to install and execute each application on each platform.]

5.3. Software Architecture

[Describe all software that is needed to support the system, and specify the physical location of all software systems. List such things as database platforms, computer languages, compilers, utilities, operating systems, communications software, programming computer-aided software engineering tools, commercial off-the-shelf (COTS) software, etc, with a brief description of the function of each item and any identifying information such as manufacturer, version number, number and types of licenses needed, etc., as appropriate. Identify all Computer Software Configuration Items (CSCIs), Computer Software Components (CSCs) and Application Programming Interfaces (APIs) to include name, type, purpose and function for each; the interfaces, messaging, and protocols for those elements; and rationale for the software architectural design. Include software modules that are functions, subroutines, or classes. Use functional hierarchy diagrams, structured organization diagrams (e.g., structure charts), or object-oriented diagrams that show the various segmentation levels down to the lowest level. All features on the diagrams should have reference numbers and names. Include a narrative that expands on and enhances the understanding of the functional breakdown. If necessary, describe how a component was further divided into subcomponents, and the relationships and interactions between the subcomponents. Proceed into as many levels/subsections of discussion as needed in order to provide a high-level understanding of the entire system or subsystem, leaving the details for inclusion in a later section of the SDD. Include data flow diagrams that conform to appropriate standards (e.g., Yourdon-Demarco conventions) and provide the physical process and data flow related to the logical process and data flow decomposed to the primitive process level (describing how each input is processed/transformed into the resulting output). If there are parts of the system that already existed before this development effort began, then only describe the relationships and interactions between the old parts and the new parts. Pre-existing parts that are modified or enhanced need to be described only to the extent that is necessary to provide a sufficient understanding of the nature of the changes that are being made.]

5.4. System Architecture Diagram

[Using the hardware, software, and communications designs described above, depict the overall, integrated structure of the system in terms of presentation, application, and data regions.]

6. Data Design

[Describe the design of all database management system (DBMS) files and non-DBMS files associated with the system. Provide a comprehensive data dictionary showing data element name, type, length, source, validation rules, maintenance (create, read, update, delete (CRUD) capability), data stores, outputs, aliases, and description. The Data Design information can be included as an appendix or recorded in a separate Database Design Document, as appropriate, which would be referenced here.]

6.1. Data Objects and Resultant Data Structures

[For each functional data object, specify the data structure(s) which will be used to store and process the data. Describe any data structures that are a major part of the system, including major data structures that are passed between components. List all functions and function parameters. For functions, give function input and output names in the description. Refer as appropriate to the decomposition diagrams.]

6.2. File and Database Structures

[Using the Logical Data Model, create a physical data model that describes data storage and manipulation in the systems architectural setting. Describe file structures and their locations. Explain how data may be structured in the selected database management system, if applicable. For networks, detail the specific distribution of data. Note any changes to the Logical Data Model which occur because of software or hardware requirements.]

6.2.1. Database Management System Files

[Provide the detailed design of the DBMS files and include the following information, as appropriate (refer to the data dictionary):

a) Refined Logical Data Model (Provide normalized table layouts, entity relationship diagrams, and other logical design information.)
b) Physical description of the DBMS schemas, sub-schemas, records, sets, tables, storage page sizes, etc.
c) Access methods (e.g., indexed, via set, sequential, random access, sorted pointer array, etc.)
d) Estimate of the DBMS file size or volume of data within the file, and data pages, including overhead resulting from access methods and free space
e) Definition of the update frequency of the database tables, views, files, areas, records, sets, and data pages (Provide an estimate of the number of transactions, if the database is an online transaction-based system.)
f) Backup and recovery specifications]

6.2.2. Non-Database Management System Files

[Provide the detailed description of all non-DBMS files and include a narrative description of the usage of each file that identifies if the file is used for input, output, or both, and if the file is a temporary file. Also provide an indication of which modules read and write the file and include file structures (refer to the data dictionary). As appropriate, the file structure information should include the following:

a) Record structures, record keys or indexes, and data elements referenced within the records

b) Record length (fixed or maximum variable length) and blocking factors
c) Access method (e.g., index sequential, virtual sequential, random access, etc.)
d) Estimate of the file size or volume of data within the file, including overhead resulting from file access methods
e) Definition of the update frequency of the file (If the file is part of an online transaction-based system, provide the estimated number of transactions per unit of time, and the statistical mean, mode, and distribution of those transactions.)
f) Backup and recovery specifications]

7. Human-Machine Interface

[Provide a description of each user class or role associated with the system. A user class is distinguished by the ways in which users interact with the proposed system or situation. Factors that distinguish a user class include common responsibilities, skill levels, work activities, and modes of interaction with the system. In this context, a user is anyone who interacts with the proposed system, including operational users, data entry personnel, system operators, operational support personnel, system maintainers, and trainers. For each user class, provide estimates of the total number of users anticipated, a maximum number of concurrent users, and the number of external users.]

7.1. Inputs

[Provide a description of the input media used by the user/operator for providing information to the system. Show a mapping to the high-level data flows (e.g., data entry screens, optical character readers, bar scanners, etc.). If appropriate, the input record types, file structures, and database structures provided in Section 6, Data Design, may be referenced. Include data element definitions, or refer to the data dictionary. Provide the layout of all input data screens or graphical user interfaces (GUIs) (e.g., windows). Define all data elements associated with each screen or GUI, or reference the data dictionary. Provide edit criteria for the data elements, including specific values, range of values, mandatory/optional, alphanumeric values, and length. Also address data entry controls to prevent edit bypassing. Discuss the miscellaneous messages associated with user/operator inputs, including the following:

a) Copies of form(s) if the input data are keyed or scanned for data entry from printed forms
b) Description of any access restrictions or security considerations
c) Each transaction name, code, and definition, if the system is a transaction-based processing system
d) Incorporation of the Privacy Act statement into the screen flow, if the system is covered by the Privacy Act
e) Description of accessibility provisions to comply with Section 508 of the Rehabilitation Act]

7.2. Outputs

[Describe the system output design relative to the user/operator. Show a mapping to the high-level data flows. System outputs include reports, data display screens and GUIs, query results, etc. The output files described in Section 6, Data Design, may be referenced. The following should be provided, if appropriate:

a) Identification of codes and names for reports and data display screens

b) Description of report and screen contents (provide a graphical representation of each layout and define all data elements associated with the layout or reference the data dictionary)
c) Description of the purpose of the output, including identification of the primary users
d) Report distribution requirements, if any (include frequency for periodic reports)
e) Description of any access restrictions or security considerations
f) Description of accessibility provisions to comply with Section 508 of the Rehabilitation Act]

8. Operational Scenarios

[Describe the general functionality of the system from the users' perspectives and provide an execution or operational flow of the system. If a separate Concept of Operations (ConOps) document exists, reference should be made to the operational scenarios contained in the ConOps. The existing ConOps should be updated, if necessary, to reflect current information or understanding. If a separate ConOps does not exist, then operational scenarios should be included here that provide step-by-step descriptions of how the proposed system should operate and interact with its users and its external interfaces under a given set of circumstances. The scenarios tie together all parts of the system, the users, and other entities by describing how they interact, and may also be used to describe what the system should not do. Operational scenarios should be described for all operational modes, transactions, and all classes of users identified for the proposed system. For each transaction, provide an estimate of the size (use maximum, if variable) and frequency (e.g., average number per session). Identify if there any transactional peak periods and include an estimate of frequency during those periods. Each scenario should include events, actions, stimuli, information, and interactions as appropriate to provide a comprehensive understanding of the operational aspects of the proposed system. The scenarios can be presented in several different ways: 1) for each major processing function of the proposed system, or 2) thread-based, where each scenario follows one type of transaction type through the proposed system, or 3) following the information flow through the system for each user capability, following the control flows, or focusing on the objects and events in the system. The number of scenarios and level of detail specified will be proportional to the perceived risk and the criticality of the project.]

9. Detailed Design

[Provide the information needed for a system development team to actually build and integrate the hardware components, code and integrate the software components, and interconnect the hardware and software segments into a functional product. Additionally, address the detailed procedures for combining separate COTS packages into a single system.]

9.1. Hardware Detailed Design

[Provide enough detailed information about each of the individual hardware components to correctly build and/or procure all the hardware for the system (or integrate COTS items). If there are many components or if the component documentation is extensive, place it in an appendix. Add additional diagrams and

information, if necessary, to describe each component and its functions adequately. Industry-standard component specification practices should be followed. For COTS components, identify specific vendor and appropriate item names and model numbers. Include the following information in the detailed component designs, as applicable:

a) Power input requirements for each component
b) Signal impedances and logic states
c) Connector specifications (serial/parallel, 11-pin, male/female, etc.)
d) Memory and/or storage space specifications
e) Processor requirements (speed and functionality)
f) Graphical representation depicting the number of hardware items (e.g., servers, I/O devices, monitors, printers, etc.), and the relative positioning of the components to each other
g) Cable type(s) and length(s)
h) User interfaces (buttons, toggle switches, etc.)
i) Hard drive/floppy drive/CD-ROM specifications
j) Monitor resolution]

9.2. Internal Communications Detailed Design

[If the system includes more than one component, there may be a requirement for internal communications to exchange information, provide commands, or support input/output functions. Provide enough detailed information about the communication design to correctly build and/or procure the communications components for the system. Include the following information in the detailed component designs, as appropriate:

a) Number of servers and clients to be included on each area network
b) Specifications for bus timing requirements and bus control
c) Format(s) for data being exchanged between components
d) Graphical representation of the connectivity between components, showing the direction of data flow (if applicable), and approximate distances between components (information should provide enough detail to support the procurement of hardware to complete the installation at a given location)
e) LAN topology]

9.3. Software Detailed Design

[Provide a detailed description for each system software component that addresses the following software component attributes. Much of the information that appears in this section should be contained in the headers/prologues and comment sections of the source code for each component, subsystem, module and subroutine. If so, this section may largely consist of references to or excerpts of annotated diagrams and source code. Any referenced diagrams or source code excerpts should be provided at any design reviews.

Component Identifier – the unique identifier and/or name of the software component.
Classification – the kind of component (e.g., subsystem, module, class, package, function, file, etc.)
Language – the programming language used for the component.
SLOC Estimate – the estimated source lines of code for the component.

Definition – the specific purpose and semantic meaning of the component.
Responsibilities – the primary responsibilities and/or behavior of the component. What does the component accomplish? What roles does it play? What kinds of services does it provide to its clients?
Requirements – the specific functional or nonfunctional requirements that the component satisfies.
Internal Data Structures – the internal data structures for the component.
Constraints – any relevant, assumptions, limitations, or constraints for the component. This should include constraints on timing, storage, or component state, and might include rules for interacting with the component (encompassing pre-conditions, post-conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)
Composition – a description of the use and meaning of the subcomponents that are a part of the component.
Users/Interactions – a description of the component's collaborations with other components. What other components is this entity used by? What other components does this entity use (including any side-effects this component might have on other parts of the system)? This includes the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated sub-classes, super-classes, and meta-classes.
Resources – a description of any and all resources that are managed, affected, or needed by the component. Resources are components external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.
Processing – a description of precisely how the component goes about performing the duties necessary to fulfill its responsibilities. This should encompass a description of any algorithms used; changes or state; relevant time or space complexity; concurrency; methods of creation, initialization, and cleanup; and handling of exceptional conditions.
Interfaces/Exports – the set of services (resources, data, types, constants, subroutines, and exceptions) that are provided by the component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include or provide a reference in its discussion to a description of its important software component attributes (Component Identifier, Classification, Language, SLOC Estimate, Definition, Responsibilities, Requirements, Internal Data Structures, Constraints, Composition, Uses/Interactions, Resources, Processing, and Interfaces/Exports).]

10. System Integrity Controls

[Provide design specifications for the following minimum levels of control and any additional controls as appropriate or necessary:

a) Internal security to restrict access of critical data items to only those access types required by users/operators
b) Audit procedures to meet control, reporting, and retention period requirements for operational and management reports
c) Application audit trails to dynamically audit retrieval access to designated critical data

d) Standard tables to be used or requested for validating data fields
e) Verification processes for additions, deletions, or updates of critical data
f) Ability to identify all audit information by user identification, network terminal identification, date, time, and data accessed or changed.]

11. External Interfaces

[Describe any interfaces that exist with external systems that are not within the scope of the system being designed, regardless whether the other systems are managed by CMS or another agency. Describe the electronic interface(s) between the system being designed and each of the other systems and/or subsystem(s), emphasizing the point of view of the system being designed. If there is more than one or two external systems, or if the interfaces are not simplistic, a separate Interface Control Document (ICD) should be prepared and referenced here.]

11.1 Interface Architecture

[Describe the interface(s) between the system being developed and other systems (e.g., batch transfers, queries, etc.). Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. If remote connectivity is required, identify the method of access. Provide a diagram depicting the communications path(s) between this system and each of the other systems, which should map to the context diagram(s) provided in Section 3, System Overview. Use subsections to address each interface independently.]

11.2 Interface Detailed Design

[For each external system with which the system being designed interfaces, describe the information exchange and rules governing the interface. Provide enough detailed information about the interface to correctly format, transmit, and/or receive data across the interface. Include the following information in the detailed design for each interface, as appropriate:

a) Data format requirements (If there is a need to reformat data before they are transmitted or after incoming data is received, include descriptions of the tools and/or methods for the reformatting process.)
b) Specifications for hand-shaking protocols between the two systems (Include the content and format of the information to be included in the hand-shake messages, the timing for exchanging these messages, and the steps to be taken when errors are identified.)
c) Format(s) for error reports exchanged between the systems (Address the disposition of error reports (e.g., retained in a file, sent to a printer, flag/alarm sent to the operator, etc.))
d) Graphical representation of the connectivity between systems, showing the direction of data flow
e) Query and response descriptions]

Appendices

[Utilize appendices to facilitate ease of use and maintenance of the SDD document. Each appendix should be referenced in the main body of the document where that information would normally have been provided. Suggested appendices include (but are not limited to):

a) **Software Architecture Diagrams** – provide the functional hierarchy diagrams, structured organization diagrams, or object-oriented diagrams that show the various segmentation levels of the software architecture down to the lowest level.
b) **Data Dictionary** – provide definitions of all processes, data flows, data elements, and data stores.
c) **Requirements Traceability Matrix** – demonstrate backward traceability of the system and software architectural designs to the functional and nonfunctional requirements documented in the Requirements Document.
d) **Software Product 508 Compliancy Checklist** – demonstrate compliance or non-compliance with accessibility standards provided in Section 508 of the Rehabilitation Act of 1973, as amended effective June 20, 2001.]

Glossary

[Provide clear and concise definitions for terms used in the SDD that may be unfamiliar to readers of the document.]