Chapter: **6.0 System Design Stage**

Description: The goal of this stage is to translate the user-oriented functional design specifications into a set of technical, computer-oriented system design specifications; and to design the data structure and processes to the level of detail necessary to plan and execute the Construction and Implementation Stages. General module specifications should be produced to define what each module is to do, but not how the module is to be coded. Effort focuses on specifying individual routines and data structures while holding constant the structure and interfaces developed in the previous stage. Each module and data structure is considered individually during detailed design with emphasis placed on the description of internal and procedural details. The primary work product of this stage is a system design that provides a blueprint for the coding of individual modules and programs.

Input: The following items provide input to this stage:

SEM Templates:

- Functional Design document
- Maintenance Plan
- Requirements Specification
- Requirements Traceability Matrix
- Software Configuration Management Plan

PMM Templates:

- Project Plan
- Quality Management Plan
- Security Plan

Other Inputs:

- Data Dictionary

High-Level Activities: The remainder of this chapter is divided into sections that describe the specific high-level activities performed during this stage. These activities represent the minimum requirements for a large information systems engineering effort. Notes are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate different sizes of information systems engineering efforts. The high-level activities are presented in the sections listed below.

6.5 Develop Conversion Plan
6.6 Develop System Design
6.7 Develop Program Specifications

Touch Points: The following touch points are involved in the System Design Stage:

- Contracts and Procurement
  - Contract Liaison involvement if contract issues arise
- E-Michigan
  - Continue to work with E-Michigan's webmaster, as appropriate, to ensure ADA compliance and Michigan.gov look and feel standards.
- Enterprise Architecture (EA)
  - Request exceptions as required
  - Complete EA Solution Assessment for the chosen solution and submit to EA for review/approval
- Infrastructure Services
  - Technical and Data Center Services Solutions Engineer involvement in completion of Hosting Solution document
  - Infrastructure Specialist involvement in establishing the construction environment
- Security
  - Review MDIT and Agency Security Policies
  - Review State and Federal Laws and Regulations
  - Revise Network Diagram and Data Flow Diagrams
  - Review Final Risk Analysis with OES recommended security controls

Output: Several work products are produced during this stage. The work products listed below are the minimum requirements for a large project. Deviations in the content and delivery of these work products are determined by the size and complexity of the project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

SEM Templates:

- Conversion Plan (*initial*)
- Maintenance Plan (*revised*)
- Requirements Traceability Matrix *(revised)*
- Software Configuration Management Plan (*final*)
- Software Testing Checklist (*final*)

- System Design Checklist (*final*)
- System Design Document (*final*)
- Test Plan *(initial)*
- Test Reports (*initial*)

PMM Templates:

- Project Plan *(revised)*
- Security Plan (*revised*)

A diagram showing the work products associated with each high-level activity is provided in *Exhibit 6.0-1, SEM Overview Diagram – System Design Stage Highlighted.*

Review the Project Plan for accuracy and completeness of all System Design Stage activities and make any changes needed to update the information.

Review Process: Quality reviews are necessary during this stage to validate the product and associated work products. The activities that are appropriate for quality reviews are identified in this chapter and Chapter 2.0, Lifecycle Model. In addition, a Critical Design Review is conducted once the System Design Document is developed. The time and resources needed to conduct the walkthroughs and Critical Design Review should be indicated in the project resources, schedule, and work breakdown structure.

Structured Walkthrough (SWT)
Requirements for a peer review or a more formal structured walkthrough are documented under *Review Process* at the end of each Task, Subtask, or Activity section in this stage. The State of Michigan guide titled *Structured Walkthrough Process Guide* provides a procedure and sample forms that can be used for SWTs. This document is available on the MDIT SUITE website.

Stage Exit
Schedule a Stage Exit as the last activity of the System Design Stage to enable the project approvers to review project deliverables and provide a concur/non-concur position to the project manager. The State of Michigan guide titled *Stage Exit Process Guide* provides a procedure and sample report form that can be used for stage exits. This document is available on the MDIT SUITE website.

References: Chapter 2.0, Lifecycle Model, *Quality Reviews* provides an overview of the Quality Reviews to be conducted on a project.

Bibliography: The following materials were used in the preparation of the System Design Stage chapter.

1. Barker, Richard, *CASE*METHOD,* Tasks and Deliverables, 1990. pp. 5-10 and 5-11.

2. Gane, Chris and Sarson, Trish, *Structured Systems Analysis: Tools and Techniques*, Prentice-Hall, Inc., Englewood Cliffs, 1979.

3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Design Descriptions,* IEEE Std 1016.1-1993, New York, 1993.

4. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications,* ANSI/IEEE Std 830-1998, New York, 1998.

5. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Recommended Practice for Software Design Descriptions,* IEEE Std 1016-1998, New York, 1998.

6. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Lifecycle Processes,* IEEE Std 1074-1991, New York, 1992.

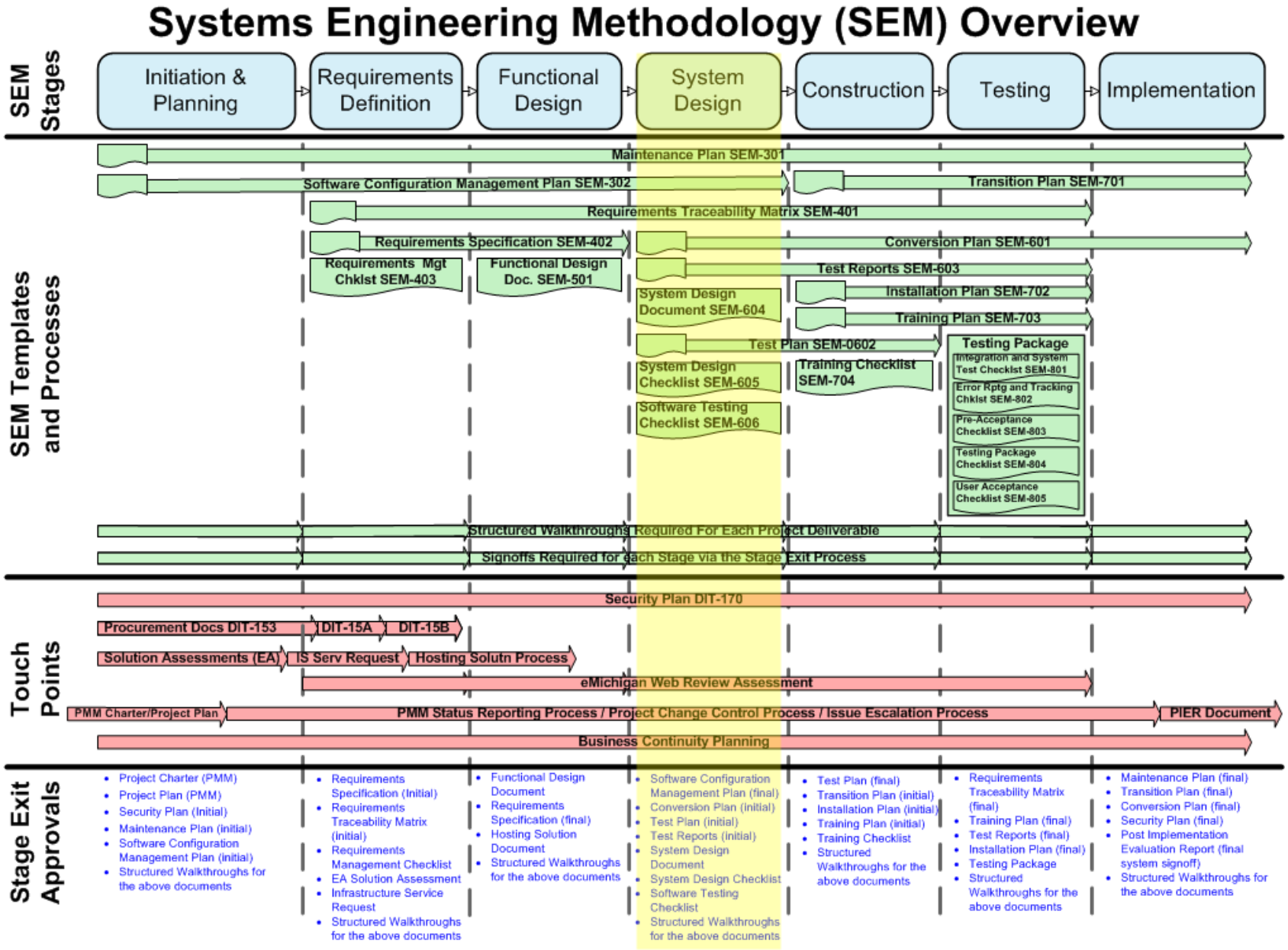
7. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans,* ANSI/IEEE Std 1012-1998, New York, 1998.

9. U.S. Department of Defense, Military Standard, *Specification Practices,* MIL-STD-490 B, 1986. pp. 47-50.

10. U.S. Department of Defense, Military Standard, *Technical Reviews and Audits for Systems, Equipments, and Computer Software,* MIL-STD-1521 B, 1985. pp.5, 33-52.

11. U.S. Social Security Administration, Office of Systems, *Software Engineering Technology (SET) Manual,* Volume 1, 1990. Part 40 Design Stage.

Exhibit 6.0-1 SEM Overview Diagram – System Design Stage Highlighted



Activity: **6.1 Design Specifications for Modules**

Responsibility: Project Team

Description: During the Functional Design Stage, a decomposition of the product requirements resulted in a collection of design entities (or objects). In the System Design Stage, these design entities are grouped into the routines, modules, and programs that need to be developed or acquired as off-the-shelf or reusable software.

Expand the functional design to account for each major action that must be performed and each data object to be managed. Detail the design to a level such that each program represents a function that a developer will be able to code.

Procedure: Use the following procedure to design the module specifications.

- Identify a program for each action needed to meet each function or data requirement in the Requirements Specification and the data dictionary.
- Identify any routines and programs that may be available as reusable code or objects from existing applications or off-the-shelf software.
- Identify programs that must be designed and developed (custom-built). Assign a name to each program and object that is functionally meaningful. Identify the system features that will be supported by each program.
- Specify each program interface. Update the data dictionary to reflect all program and object interfaces changed while evolving from the functional to the system design.
- Define and design significant attributes of the programs to be custom-built.
- Expand the program interfaces to include control items needed for design validity (e.g., error and status indicators).
- Combine similar programs and objects. Group the design entities into modules based on closely knit functional relationships. Formulate identification labels for these modules.
- Show dependencies between programs and physical data structures (e.g., files and global tables). Avoid defining a program that not only needs data residing in a file or global table, but also depends on the physical structure or location of data.

- Change the design to eliminate features that reduce maintainability or reusability (i.e., minimize coupling between programs and maximize the cohesion of programs).

Work Product: Document the system design primarily in the form of diagrams. Supplement each diagram with text that summarizes the function (or data) and highlights important performance and design issues.

When using structured design methods, the design diagrams should:

- Depict the product as a top-down set of diagrams showing the control hierarchy of all programs to be implemented.
- Define the function of each program.
- Identify data and control interfaces between programs.
- Specify files, records, and global data accessed by each program.

When using object-oriented or data-centered design methods, the design diagrams should:

- Show the data objects to be managed by the product.
- Specify the program functions to be included within each object.
- Identify functional interfaces between objects.
- Specify files and records comprising each object.
- Identify relationships between data files.

Review Process: Conduct structured walkthroughs to assure that the custom-built routines and programs are correctly designed.

Activity: **6.2 Design Physical Model and Database Structure**

Responsibility: Project Team

Description: The physical model is a description of the dynamics, data transformation, and data storage requirements of the product. The physical model maps the logical model created during the Functional Design Stage to a specific technical reality. Care must be taken to retain in the physical implementation all of the capabilities inherent in the logical model.

The physical model frequently differs from the logical model in the following areas.

- Constraints imposed by the database management system - The logical data model may have different implementations in the selected database management system.
- Performance - Data redundancies, indices, and data structure changes may have to be introduced into the physical model to improve performance.
- Distributed processing - Possible network and multiple production hardware configurations may cause changes to the physical data model.

Designing the database structure converts the data requirements into a description of the master and transient files needed to implement the requirements. If the product will include a database, design the database in conjunction with the following database management features.

- Report writer and file processing capabilities
- Online query processing to retrieve data
- Automated data dictionary systems

Work Product: Document the physical model for incorporation into the System Design Document. Review the contents of the data dictionary entries and update to complete information on data elements, entities, files, physical characteristics, and data conversion requirements.

Review Process: Schedule structured walkthroughs to verify that the physical model and data dictionary are correct and complete.

Activity: **6.3 Develop Integration Test Considerations**

Responsibility: Project Team Developers

Description: The purpose of integration testing is to verify the integrity of a module (a cohesive set of programs) and its interfaces with other modules within the product structure. An integration test plan is developed to incorporate successfully unit-tested modules into the overall structure and to test each level of integration to isolate errors introduced by newly incorporated modules.

The number of integration levels, the classes of tests to be performed, and the order in which routines and builds are incorporated into the overall structure are addressed in the Integration Test portion of the overall Test Plan package. The following factors should be considered.

- Are routines to be integrated in a pure top-down manner or should builds be developed to test sub-functions first?
- In what order should major functions be incorporated?
- Is the scheduling of module coding and testing consistent with the order of integration?
- Is special hardware required to test certain routines?

Integration testing should include tests that validate the following functions.

- Verify each interface between the module and all other modules.
- Access each input message or command processed by the module.
- Check each external file or data record referenced by coding statements in the module.
- Output each message, display, or record generated by the module.

An important consideration during integration test planning is the amount of test software (e.g., drivers, test case generation) that must be developed to adequately test the required functionality. For example, it may be cost-effective to delay testing of a communication function until hardware is available rather than generate test software to simulate communication links.

Similarly, it may be better to include certain completed modules in the structure in order to avoid having to develop software drivers. These decisions are made on the basis of cost and risks.

Work Product: Develop draft Integration Testing procedures that address the following activities.

- Define the integration tests at each element level, stating objectives, what is to be tested, and verified. Testing is from the point of view of structure and function.
- Define all aspects of the formal interfaces that must undergo formal integration testing. Review interface requirements to ensure completeness, consistency, and effectiveness.
- Plan for test tools and software that must be developed to adequately test the required functionality.

Note: The Integration Test Considerations are incorporated in the Test Plan.

Review Process: Conduct a peer review or structured walkthrough to assure that the draft Integration Test section of the Test Plan is accurate and complete. The Integration Test section will be reviewed and revised as needed during the Construction Stage.

Activity: **6.4 Develop System Test Considerations**

Responsibility: Project Test Team

Description: The objectives of the system test process are to assure that the product adequately satisfies the project requirements; functions in the computer operating environment; successfully interfaces between user procedures, operating procedures, and other systems; and protects the software and data from security risks. The system should be tested under the same kind of daily conditions that will be encountered during regular operations. System timing, memory, performance, and security functions are tested to verify that they perform as specified. The functional accuracy of logic and numerical calculations are tested for verification under normal and load conditions.

Test data should be varied and extensive enough to enable the verification of the operational requirements. Expected output results should be included in the test plan in the form of calculated results, screen formats, hardcopy output, predetermined procedural results, warnings, error messages and recovery.

Detailed planning for the system testing helps to ensure that system acceptance will be successfully completed on schedule. The Software Testing Checklist should be completed during this process. When applicable, system testing must include the following types of tests.

- Performance tests that measure throughput, accuracy, responsiveness, and utilization under normal conditions and at the specified maximum workload.
- Stress tests to determine the loads that result in appropriate, non-recoverable, or awkward system behavior.
- Interface tests to verify that the system generates external outputs and responds to external inputs as prescribed by approved interface control documentation.
- Infrastructure and system recovery and reconfiguration tests.
- Verification that the infrastructure and system can be properly used and operated in accord with user's guides and operating instructions.
- Verification that the infrastructure and system meet their requirements for reliability, maintainability, and availability, including fault tolerance and error recovery.

- Verification of the effectiveness of error detection and analysis, and automated diagnostic tools.

- Demonstration that the infrastructure and system comply with their serviceability requirements such as accessibility, logistics, upgrades, diagnostics, and repair capabilities.

Work Product: Complete the Software Testing Checklist to ensure that all testing considerations have been addressed.

Develop draft System Test Considerations that describe the testing effort, provide the testing schedule, and define the complete range of test cases that will be used to assure the reliability of the product. The test cases must be complete and the expected output known before testing is started. The test considerations should address the following.

- Provide a definition of, and the objectives for, each test case.

- Define the test scenario(s) including the step-by-step procedure, the number of processing cycles to be tested or simulated, and the method and responsibility for feeding test data to the system.

- Define the test environment including the infrastructure and software environment under which the testing will be conducted.

- Identify and describe manual procedures, automated procedures, and test sites (real or simulated).

- Identify test tools and special test support needs (e.g., hardware and software to simulate operational conditions or test data that are recordings of live data).

- Identify responsibilities for conducting tests; for reviewing, reporting, and approving the results; and for correcting error conditions.

- Develop a requirements verification matrix mapping individual tests to specific requirements and specifying how each system requirement will be validated.

- Schedule for integrating and testing all components including adequate time for retesting.

Note: The System Test Considerations are incorporated into the Test Plan.

Review Process: Conduct peer reviews or structured walkthroughs to assure that each system test procedure is accurate, complete, and accomplishes the stated objectives. The System Test Considerations will be reviewed and revised as needed during the Construction Stage.

Activity: **6.5 Develop Conversion Plan**

Responsibility: Project Team

Description: If the product will replace an existing IT system, develop a Conversion Plan. The major elements of the Conversion Plan are to develop conversion procedures, outline the installation of new and converted files/databases, coordinate the development of file-conversion, and plan the implementation of the conversion procedures.

File conversion should include a confirmation of file integrity. Determine what the output in the new system should be compared with the current system, and ensure that the files are synchronized. The objective of file conversion is new files that are complete, accurate and ready to use.

Many factors influence data conversion, such as the design of the current and new systems and the processes for data input, storage, and output. Understanding the data's function in the old system and determining if the function will be the same or different in the new system is of major importance to the Conversion Plan. The structure of the data to be converted can limit the development of the system and affect the choice of software.

Work Product: Develop a Conversion Plan that identifies what conversions are needed and how the conversion(s) will be implemented. Consider the following factors during the development of the Conversion Plan.

- Determine if any portion of the conversion process should be performed manually.
- Determine whether parallel runs of the old and new systems will be necessary during the conversion process.
- Understanding the function of the data in the old system and determining if the use will be the same or different in the new system is important.
- The order that data is processed in the two systems influences the conversion process.
- Volume considerations, such as the size of the database and the amount of data to be converted, influence how the data will be converted. Especially important are the number of reads that are necessary and the time these conversions will take. The Hosting Solution Document provides guidance in assessing database factors.

- User work and delivery schedules, timeframes for reports and end-of-year procedures, and the criticality of the data help determine when data conversion should be scheduled.

- Determine whether data availability and use should be limited during the conversion.

- Plan for the disposition of obsolete or unused data that is not converted.

Review Process: Conduct structured walkthroughs to assure that the Conversion Plan is accurate and complete.

Resource: A Conversion Plan template is available on the MDIT SUITE website.

Activity: **6.6 Develop System Design**

Responsibility: Project Team

Description: The system design is the main technical work product of the System Design Stage. The system design translates requirements into precise descriptions of the components, interfaces, and data necessary before coding and testing can begin. It is a blueprint for the Construction Stage based on the structure and data model established in the Functional Design Stage.

The system design plays a pivotal role in the development and maintenance of a product. The design provides valuable information used by the project manager, quality assurance staff, software configuration management staff, software designers, developers, testers, and maintenance personnel.

The system design is baselined after the system owner's formal approval of the design as described in the System Design Document. Once the system design is baselined, any changes to the design must be managed under change control procedures established in the Software Configuration Management Plan. Approved changes must be incorporated into the System Design Document.

It is important for the system owner/users to understand that some changes to the baselined system design may affect the project scope and therefore can change the project cost, resources, or schedule. It is the responsibility of the project manager and team to identify system owner/user requested changes that would result in a change of project scope; evaluate the potential impact to the project costs, resources, or schedule; and notify the system owner of the project planning revisions that will be required to accommodate their change requests.

Work Product: Major work products include the System Design Document and the updated Requirements Traceability Matrix. Each requirement identified in the Requirements Specification must be traceable to one or more design entities. This traceability ensures that the product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Revise the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the system design to the requirements.

Review Process: Conduct a structured walkthrough of the System Design Document and the Requirements Traceability Matrix.

Refer to task 6.6.2, *Conduct System Design Review*, for the system design review process.

Tasks: The following tasks are involved in developing the system design.

6.6.1 Develop System Design Document
6.6.2 Conduct System Design Review

Task: **6.6.1 Develop System Design Document**

Description: The System Design Document records the results of the system design process and describes how the product will be structured to satisfy the requirements identified in the Requirements Specification. The System Design Document is a translation of the requirements into a description of the structure, components, interfaces, and data necessary to support the construction process.

Work Product: Prepare the System Design Document and submit it to the system owner and users for their review and approval. The approved System Design Document is the official agreement and authorization to use the design to build the product. Approval implies that the design is understood, complete, accurate, and ready to be used as the basis for the subsequent lifecycle stages. In other words, once approved this becomes the design baseline. Subsequent changes or additions to the design that receive stakeholder concurrence supersede the existing baseline and establish a new design baseline. Place a copy of the approved System Design Document in the Project File.

Review Process: Conduct structured walkthroughs as needed to ensure that the System Design Document is accurate and complete.

Resource: A System Design Document is available on the MDIT SUITE website.

Task: **6.6.2 Conduct System Design Review**

Description: The System Design Review is a formal technical review of the system design. The purpose of the review is to demonstrate to the system owner and users that the system design can be implemented on the selected platform and accounts for all software and data requirements and accommodates all design constraints (e.g., performance, interface, security, safety, resource, and reliability requirements). The design review should include a review of the validity of algorithms needed to perform critical functions.

Several short System Design Reviews can replace one long review if the product consists of several components that are not highly interdependent. The review process should be a series of presentations by the project team to the system owner and other approval authorities.

Using the System Design Checklist, conduct a System Design Review that demonstrates that the design specifications are capable of supporting the full functionality of the product, as follows:

- All algorithms will perform the required functions.
- The specification is complete, unambiguous and well documented, including timing and sizing, and data and storage allocations.
- The specification is necessary and sufficient for, and directly traceable to, the system design.
- The specification is compatible with every other specification, piece of equipment, facility, and item of system architecture, especially as regards information flow, control, and sequencing.
- The specification is consistent with the abilities of current development and user personnel.

In addition to verifying individual specifications, the System Design Review assesses other project work products to ensure the following.

- The approved design approach is being followed by the team.
- Measures to reduce risk on a technical, cost, and schedule basis are adequate.
- The performance characteristics of the design solution are acceptable.
- Testing will be sufficient to ensure product correctness.

- The resultant application will be maintainable.
- Provisions for automatic, semi-automatic, and manual recovery from hardware/software failures and malfunctions are adequate and documented.
- Diagnostic programs, support equipment, and commercial manuals all comply with the system maintenance concept and specification requirements.

Work Product: Complete the System Design Checklist as part of the System Design review process.

Create and distribute official meeting minutes for each design review session. The minutes should consist of significant questions and answers, action items and individual/group responsible, deviations, conclusions, and recommended courses of action resulting from presentations or discussions. Recommendations that are not accepted should be recorded along with the reason for non-acceptance. Minutes must be distributed to review participants. The system owner determines review performance as follows:

- Approval - The review was satisfactorily completed.
- Contingent Approval - The review is not finished until the satisfactory completion of identified action items.
- Disapproval - The specification is inadequate. Another System Design Review will be required, once the required changes are made to the system design specifications.

Activity: **6.7 Develop Program Specifications**

Responsibility: Project Team

Description: A Program Specification is a written procedural description of each system routine. The Program Specification should provide precise information needed by the developers to develop the code.

Many techniques are available for specifying the system design, such as formal specification languages, program design languages (e.g, pseudo-code or structured English), meta-code, tabular tools (e.g., decision tables), and graphical methods (e.g., flow charts or box diagrams). In object-oriented design, the specification of requirements and preliminary design constraints and dependencies often results in the design language producing the detailed specifications.

Select the technique or combination of techniques that is best suited to the project and to the experience and needs of the developers who will use the system design as their blueprint. The following are suggestions for using the techniques.

- Decision trees are useful for logic verification or moderately complex decisions that result in up to 10-15 actions. Decision trees are also useful for presenting the logic of a decision table to users.

- Decision tables are best used for problems involving complex combinations of up to 5-6 conditions. Decision tables can handle any number of actions; however, large numbers of combinations of conditions can make decision tables unwieldy.

- Structured English is best used wherever the problem involves combining sequences of actions with decisions or loops. Once the main work of physical design has been done and physical files have been defined, it becomes extremely convenient to be able to specify physical program logic using the conventions of structured English, but without getting into the detailed syntax of any particular development language (pseudo-code).

- Standard English is best used for presenting moderately complex logic once the analyst is sure that no ambiguities can arise.

Work Product: Specifications may be produced as documents, graphic representations, formal design languages, records in a database management system, and CASE tool dictionaries. A list of program attributes typically included in a Program Specification is provided at the end of this section.

Review Process: Conduct a series of structured walkthroughs to ensure that the Program Specification is accurate and complete.

Sample Attributes: For each program to be custom built, define the program's functional and technical attributes as they become known. The following is a list of sample program attributes.

- Program identification
- Program name
- Program generic type
- Functional narrative
- Program hierarchical features diagram
- Development dependencies and schedule
- Operating environment
    - equipment
    - development language and version
    - preprocessor
    - operating system
    - storage restrictions
    - security
- Frequency of run
- Data volumes
- Program termination messages
    - normal termination
    - abnormal termination
- Console/printer messages
- Recovery/restart procedures
- Software objectives
- Program input/output diagram
- Data bank information
- Called and calling programs/modules
- Program logic diagrams
- Significant "how-to" instructions
- Telecommunications information