Software Engineering Methodology

Chapter 6.0
System Design Stage

Table of Contents

Chapter **Page**

6.0 System Design Stage 6.0-1
6.1 Select System Architecture 6.1-1
6.1.1 Evaluate System Architecture Alternatives 6.1-2
6.1.2 Recommend System Architecture 6.1-5
6.2 Design Specifications for Software Modules 6.2-1
6.3 Design Physical Model and Data Base Structure 6.3-1
6.4 Develop Integration Test Plan 6.4-1
6.5 Develop System Test Plan 6.5-1
6.6 Develop Conversion Plan 6.6-1
6.7 Develop System Design 6.7-1
6.7.1 Develop System Design Document 6.7-2
6.7.2 Conduct Critical Design Review 6.7-3
6.8 Develop Program Specifications 6.8-1
6.9 Define Programming Standards 6.9-1
6.10 Revise Project Plan 6.10-1
6.11 Conduct In-Stage Assessment 6.11-1
6.12 Conduct System Design Stage Exit 6.12-1

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 Programming and Installation 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 software 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 software system design that provides a blueprint for the coding of individual modules and programs.

Input: The following items provide input to this stage.

- Project File
- Design records
- Logical model
- Data dictionary *(expanded)*
- Requirements Traceability Matrix *(expanded)*
- Functional Design Document
- Hardware and software procurement records
- Project Plan *(revised)*
- Software Quality Assurance Plan

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 software engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate different sizes of software engineering efforts. The high-level activities are presented in the sections listed below.

6.1 Select System Architecture
6.2 Design Specifications for Software Modules
6.3 Design Physical Model and Data Base Structure
6.4 Develop Integration Test Plan
6.5 Develop System Test Plan
6.6 Develop Conversion Plan
6.7 Develop System Design

High-Level

Activities,continued:

6.8 Develop Program Specifications
6.9 Define Programming Standards
6.10 Revise Project Plan
6.11 Conduct In-Stage Assessment
6.12 Conduct System Design Stage Exit

Output:

Several work products are produced during this stage. The work products listed below are the minimum requirements for a large software 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.

- Design specifications
- Physical Model
- Data Dictionary *(expanded)*
- Integration Test Plan *(draft)*
- System Test Plan *(draft)*
- Conversion Plan
- Requirements Traceability Matrix *(expanded)*
- System Design Document
- Program Specifications
- Programming Standards
- Project Plan *(revised)*

A matrix showing the work products associated with each high-level activity is provided in *Exhibit 6.0-1, System Design Stage Activities and Work Products by Project Size.* The matrix also shows which work products are deliverables and whether they are required or optional for small, medium, and large projects.

Review Process:

Structured walkthroughs are necessary during this stage to validate work products. The activities that are appropriate for structured walkthroughs are identified throughout the chapter. In addition, a Critical Design Review is conducted once the System Design Document is developed. This review is an important milestone in the design process. 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.

Reference:

Appendix C, Conducting Structured Walkthroughs, provides a procedure and sample forms that can be used for structured walkthroughs.

Exhibit 6.0-1. System Design Stage Activities and Work Products by Project Size

Work Activity		Project Size			Work Product	Scheduled Deliverables		
		L	M	S		L	M	S
6.1	Select System Architecture	A	A	A	Analysis of Benefits and Costs Report Summary and recommendations of architecture alternatives	A A	A A	A A
6.2	Design Specifications for Software Modules	R	R	R	Design diagrams with text	R	R	R
6.3	Design Physical Model and Data Base Structure	R	R	R	Data Dictionary *(expanded)* Physical Model	R R	R R	R R
6.4	Develop Integration Test Plan	R	R	R	Integration Test Plan *(draft)*	R	R	R
6.5	Develop System Test Plan	R	R	R	System Test Plan *(draft)*	R	R	R
6.6	Develop Conversion Plan	A	A	A	Conversion Plan	A	A	A
6.7	Develop System Design	R	R	R	Requirements Traceability Matrix *(expanded)* System Design Document Critical Design Review minutes	R R R	R R R	R R R
6.8	Develop Program Specifications	R	R	R	Program Specifications	R	R	R
6.9	Define Programming Standards	R[2]	R[2]	0	Programming Standards	R[2]	R[2]	R[2]
6.1	Revise Project Plan	R	R	R	Project Plan *(revised)*	R	R	R
6.11	Conduct In-Stage Assessment	R	R	A	ISA Report Form[1]	N	N	N
6.12	Conduct System Design Stage Exit	R	R	A	Stage Exit Meeting Summary	N	N	N

Size: L = Large
M = Medium
N = Not Applicable

Minimum Requirements: R = Required
A = As Appropriate

[1] = Completed by reviewer
[2] = Can use existing plan/procedure

S= Small

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. Bramucci, Wilma, "Systems Development Software," *Faulkner Technical Reports, Inc.*, June 1989. pp. 1-7.

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

4. *Software Engineering Handbook* Chapter 5, Software Design; and Chapter 7, Software Testing.

5. *Software System Testing and Quality Assurance*, Chapter 5, Integration.

6. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications,* ANSI/IEEE Std 830-1984, New York, 1984.

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

8. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Recommended Practice for Software Design Descriptions,* IEEE Std 1016-1987, New York, 1987.

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

10. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans,* ANSI/IEEE Std 1012-1986, New York, 1986.

11. U.S. Department of Commerce, National Bureau of Standards, *Guideline for Planning and Management of Database Applications,* Federal Information Processing Standards Publication 77, 1980. pp. 10-23.

12. U.S. Department of Defense, Military Standard, *Defense System Software Development,* MIL-STD-2167A, 1988. pp. 27-29.

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

14. 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.

15. U.S. Department of Energy, *Automated Data Processing Systems Development Methodology, Volume 1,* K/CSD/INF/86-3, Vol.1,R3, prepared under contract by Martin Marietta Energy Systems, Inc. at Oak Ridge National Laboratory, August 1987.

16. U.S. Department of Energy, *Hanford Site Data Administration Guide,* RLO-ADP-1, July 1989.

Bibliography, continued:

17. U.S. Department of Energy, *Software Management Guide,* DOE/AD-0028, 1992.

18. U.S. Department of Energy, Nevada Operations Office, *Software Management Plan,* May 1991.

19. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Concepts and Procedures Manual,* 1988.

20. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Standards Manual*, 1988.

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

22. Yourdon, inc., *Management Overview of Real-Time Structured Analysis and Design*, Edition 3.0, January 1984.

Activity: **6.1**

Select System Architecture

Responsibility: Project Team

Description: When the system architecture for the software product has not been predetermined by the existing computing environment of the system owner and users, evaluate system architecture alternatives to determine which one has the best, cost-effective solution that satisfies the project requirements.

"Cost effective solution" does not imply the least expensive alternative. The "best, cost effective solution" is the alternative that does the best job of satisfying the project requirements, assures the highest quality software product, and provides for an adequate return on investment in a timeframe that is acceptable to the system owner.

Select the specific hardware, software, data base management system, and communication facilities based on the following types of considerations.

- Departmental or site-specific information architecture guidelines or standards
- Hardware and software that emphasizes simplicity, flexibility, ease of operation and maintenance
- Cost to procure and maintain potential environment
- Backup and recovery procedures
- Selection of a distributed or centralized processing environment
- Communication requirements
- Data configuration

Obtain support from functional area points-of-contact to aid in the architecture evaluation process. Consultations and input may be helpful from system and data base administrators, local area network administrators, operations personnel, system programmers, and telecommunication experts.

The following tasks are involved in selecting a system architecture.

6.1.1 Evaluate System Architecture Alternatives
6.1.2 Recommend System Architecture

Task: **6.1.1**
Evaluate System Architecture Alternatives

Description: Consider system architecture alternatives within the site's information architecture guidelines that enable the project objectives and requirements to be achieved. The selection of a system architecture depends on many factors such as the experience of the project team with each alternative and the availability of reusable components to facilitate the implementation of an alternative.

When investigating alternatives, consider the following issues.

- Those functions or portions of functions that are to be automated and the functions that will be manual. Conduct an examination of *what* the automated portion of the project will encompass.

- The technical solution for the objectives. The determinations of *how* the software product is to be designed; (e.g., online vs. batch, client-server vs. mainframe, Oracle vs. Sybase).

- The system owner's and users' computing environment and the needs created by the technical solution. Consider any hardware and software that must be acquired, including system access software, operating system software, data base management system, and telecommunications facilities.

Procedure: The following procedure provides one approach for evaluating the architecture alternatives.

- Conduct an Analysis of Benefits and Costs to determine the most cost effective alternative. On the benefits side, include the improvements over the current process being used to support the business application. On the costs side, include any degradation from current capabilities along with the rationale for allowing the degradation.

- Create and evaluate a data flow diagram for each alternative.

- Identify how users would interact with the features associated with each alternative (such as the generation of queries and reports).

- Create a list of the risks associated with each alternative and develop a plan for mitigating each risk.

Procedure, continued:

- Compare the performance capabilities of each alternative. How fast will each alternative be able to process the user's work given a particular hardware resource. Performance is usually expressed in terms of throughput, run time, or response time. Five factors that frequently affect performance include:

    - Number of intermediate files in a system (park data between programs)
    - Number of times a given file is passed
    - Number of seeks against a disk file
    - Time spent in calling programs and other system overhead
    - Time taken to execute actual program

- Compare the security and access control features of each alternative. To what extent does the alternative provide security against human errors, machine malfunction, or deliberate mischief. Some common controls include:

    - Check digits on predetermined numbers
    - Batch control totals
    - Creation of journals and audit trails
    - Limited access to files

- Compare the ease with which each alternative allows the system to be modified to meet changing requirements, such as:

    - Fixing errors
    - Changing user needs
    - Mandatory/statutory modifications
    - Enhancements

Work Product: Maintain records on each alternative that is evaluated. Use this information to develop a summary of the system architecture alternatives. The summary will be integrated into the materials presented to the system owner when a system architecture recommendation is made. Place a copy of the records for each alternative and the summary in the Project File.

Work Product continued: If an Analysis of Benefits and Costs (ABC) is conducted, prepare a report that describes the process used for the analysis, a summary of the alternatives considered, and the results obtained. The report will be integrated into the materials presented to the system owner when a system architecture recommendation is made. Place a copy of the ABC records and report in the Project File.

References: The following documents provide detailed guidance on conducting an Analysis of Benefits and Costs.

- *Analysis of Benefits and Costs (ABC's) Guideline. Volume 1, A Manager's Guide to Analysis of Benefits and Costs.* U.S. Department of Energy.

- *Analysis of Benefits and Costs (ABC's) Guideline. Volume 2, An Analyst's Handbook for Analysis of Benefits and Costs.* U.S. Department of Energy.

Task: **6.1.2**
Recommend System Architecture

Description: Based on the results of the architecture alternatives evaluation, develop a recommendation for a system architecture that is cost-effective and will facilitate the achievement of the software project requirements. Prepare a presentation for the system owner and users that provides the following types of information to support the recommendation.

- Review the limitations or problems with any current manual or automated system that will be resolved by the software product.
- Present the logical model for the software product. Highlight new functions that would be incorporated.
- For each architecture alternative that was evaluated, present the following information.
    - A description of the alternative.
    - An overall data flow diagram showing how the alternative would be implemented.
    - The way the system would look to the users, in terms of hardware, user interface, reports, and query facilities.
    - The estimated benefits of the alternative.
    - The estimated cost and time to implement the alternative.
    - A statement of the element of risk associated with the alternative.
- Present the recommended alternative and explain why it was selected.

Before the project proceeds, the system owner should make a decision about the system architecture either by formally accepting the project team's recommendation or by directing the team to use a different architecture. Any delay in making this decision could result in a slippage of the project schedule.

Work Product: Document the project team's recommendation for the most cost-effective and viable architecture alternative. Provide a summary of each alternative that was evaluated. Describe the rationale for proposing the recommended architecture. Describe the impact of this alternative on the system owner and users organization(s) and other systems. Include any background information that was relevant to the decision process, such as the Analysis of Benefits and Costs Report.

Present the project team's recommendation for the system architecture to the system owner and users. The recommendation can be delivered as a document or as a presentation. Place a copy of the document or presentation materials in the Project File.

Review Process: Conduct a structured walkthrough to assure that the most cost-effective and viable architecture alternative is being recommended.

Activity: **6.2**
Design Specifications for Software Modules

Responsibility: Project Team

Description: During the Functional Design Stage, a decomposition of the software 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 software 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 programmer will be able to code.

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

- Identify a software program for each action needed to meet each function or data requirement in the Software 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. The System Review Inventory System (SRIS) maintained at DOE Headquarters and the Energy Science and Technology Software Center (ESTSC) located at Oak Ridge, Tennessee are recommended sources for identifying reusable software. The ESTSC is the Department's central collection of DOE-supported software packages. The Center also collects software from the Nuclear Regulatory commission and others, and maintains contact with other software centers.

- 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.

Procedure, continued:

- 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 and 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 software as a top-down set of diagrams showing the control hierarchy of all software programs to be implemented.

- Define the function of each software 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 software.

Work Product, continued:

- 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: **6.3**
Design Physical Model and Data Base Structure

Responsibility: Project Team

Description: The physical model is a description of the dynamics, data transformation, and data storage requirements of the software 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 data base management system - The logical data model may have different implementations in the selected data base 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 data base structure converts the data requirements into a description of the master and transient files needed to implement the requirements. If the software product will include a data base, design the data base in conjunction with the following data base 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. Place a copy of all physical model and data base structure records in the Project File.

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

Activity: **6.4**

Develop Integration Test Plan

Responsibility: Project Team Programmers

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 software structure. An integration test plan is developed to incorporate successfully unit-tested modules into the overall software 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 software structure are addressed in the Integration Test Plan. The following factors should be considered.

- Are routines to be integrated in a pure top-down manner or should builds be developed to test subfunctions first?
- In what order should major software 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.

Description, continued: 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 software structure in order to avoid having to develop software drivers. These decisions are made on the basis of cost and risks.

Work Product: Develop the draft Integration Test Plan that addresses 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.

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

Activity: **6.5**
Develop System Test Plan

Responsibility: Project Test Team

Description: The objectives of the system test process are to assure that the software 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. 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.

- System recovery and reconfiguration tests.

- Verification that the system can be properly used and operated in accord with its users guide and operating instructions.

Description, continued:

- Verification that the system meets its 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 system complies with its serviceability requirements such as accessibility, logistics, upgrades, diagnostics, and repair capabilities.

Work Product: Develop a draft System Test Plan that describes the testing effort, provides the testing schedule, and defines the complete range of test cases that will be used to assure the reliability of the software. The test cases must be complete and the expected output known before testing is started. The test plan 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 hardware 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.

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 Plan will be reviewed and revised as needed during the Programming Stage.

Activity: **6.6**
Develop Conversion Plan

Responsibility: Project Team

Description: If the software product will replace an existing automated 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/data bases, coordinate the development of file-conversion programming, 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.

Work Product, continued:

- Volume considerations, such as the size of the data base 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.

- 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.

Activity: **6.7**
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 software components, interfaces, and data necessary before coding and testing can begin. It is a blueprint for the Programming Stage, based on the software structure and data model established in the Functional Design Stage.

The system design plays a pivotal role in the development and maintenance of a software product. The design provides valuable information used by the project manager, quality assurance staff, configuration management staff, software designers, programmers, 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: Each requirement identified in the Software Requirements Specification must be traceable to one or more design entities. This traceability ensures that the software product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the system design to the requirements. Place a copy of the expanded matrix in the Project File.

The following tasks are involved in developing the system design.

6.7.1 Develop System Design Document
6.7.2 Conduct Critical Design Review

Task: **6.7.1**
Develop System Design Document

Description: The System Design Document records the results of the system design process and describes how the software product will be structured to satisfy the requirements identified in the Software Requirements Specification. The System Design Document is a translation of the requirements into a description of the software structure, software components, interfaces, and data necessary to support the programming 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 software product. Approval implies that the design is understood, complete, accurate, and ready to be used as the basis for the subsequent lifecycle stages. 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.

The completion of the System Design Document is an appropriate time to schedule an In-Stage Assessment (ISA). The *In-Stage Assessment Process Guide* provides a description and instructions for conducting an ISA. A copy of the guide is provided in Appendix D.

Task: **6.7.2**
Conduct Critical Design Review

Description: The Critical 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 Critical Design Reviews can replace one long review if the software 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.

Conduct a Critical Design Review that demonstrates that the design specifications are capable of supporting the full functionality of the software 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 software 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 Critical 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.

Description, continued:

- The performance characteristics of the design solution are acceptable.

- Testing will be sufficient to ensure software 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: 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 resultant action items.

- Disapproval - The specification is inadequate. Another Critical Design Review will be required.

Activity: **6.8**
Develop Program Specifications

Responsibility: Project Team

Description: A Program Specification is a written procedural description of each software system routine. The Program Specification should provide precise information needed by the programmers 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 software project and to the experience and needs of the programmers 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 programming 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 data base management system, and CASE tool dictionaries. A list of significant 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 sample list of program attributes.

- Program identification
- Program name
- Program generic type
- Functional narrative
- Program hierarchical features diagram
- Development dependencies and schedule
- Operating environment
  - equipment
  - programming 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

Activity: **6.9**
Define Programming Standards

Responsibility: Project Team Programmers

Description: Programming standards are necessary to ensure that custom-built software has acceptable design and structural properties. Programming standards must be practical, easy to implement, and accepted by the project team. The project team programmers should be the primary developers of the standard. Use a structured approach to programming to allow for easy modification and to facilitate testing and debugging.

The following guidelines are generally applicable to any programming language. Use these guidelines as the basis for the programming standard and add project-specific standards relating to the programming language and tools.

- Control Flow Constructs
  - sequence
  - if-then-else
  - case statement
  - do-while (pretest loop)
  - do-until (post-test loop)
- Module Size
  - Number of executable lines of source code should average 100 lines per unit.
  - Units should contain no more than 200 lines of executable source code.
- Module Design
  - Units do not share temporary storage locations for variables
  - Units perform a single function
  - Avoid self-modifying code
  - Each unit is uniquely named
  - Each unit has a standard format:
    prologue
    variable declarations
    executable statements/comments
  - Use single entry/exit points except for error paths
  - Set comments off from the source code in a uniform manner

Description,

continued:

- Symbolic Parameters
  - Use instead of specific numerics
  - Use for constants, size of data structures, relative position in list
- Naming Conventions
  - Use uniform naming throughout each unit and module to be put under configuration control
  - Use meaningful variable names
  - Do not use keywords as identifiers
- Mixed Mode Operations
  - Avoid mixed mode expressions
  - Add comments in code whenever used
- Error and Diagnostic Messages
  - Design messages to be self-explanatory and uniform
  - Do not require user to perform table lookups
- Style
  - Use conventions such as indentation, white space, and blank lines to enhance readability
  - Align compound statements
  - Avoid "goto" statements.
  - Avoid compound, negative Boolean expressions
  - Avoid nesting constructs beyond five levels deep
  - Avoid deeply nested "if" statements.
  - Use parentheses to avoid ambiguity
  - Include only one executable statement per line
  - Avoid slick programming tricks that may create or encourage defects or be difficult to maintain; the most direct solution is best

Work Product: Create a programming standards document and distribute the document to all project team members. An existing programming standard can be used if it is applicable to the programming language and tools being used for the project.

Review Process: Conduct a peer review to assure that the programming standards are complete and appropriate for the project's programming language and tools.

Activity: **6.10**
Revise Project Plan

Responsibility: Project Manager

Description: Once the Critical Design Review is completed, the system design is baselined, and the work products from the Functional Design Stage have been updated as needed to reflect changes caused by the system design, determine if the project estimates for resources, cost, and schedule need to be revised.

Work Product: Review the Project Plan for accuracy and completeness of all System Design Stage activities and make any changes needed to update the information. Expand the information for the Programming Stage to reflect accurate estimates of resources, costs, and hours. Place a copy of the revised Project Plan in the Project File.

Note: A Project Plan is an effective management tool that is recommended for all projects regardless of size. The plan can be consolidated for small projects.

Review Process: Conduct a structured walkthrough to ensure that the Project Plan reflects the project's current status and adequately estimates the resources, costs, and schedule for the Programming Stage.

The Project Plan is formally reviewed during the In-Stage Assessment and Stage Exit processes.

Activity: **6.11**
Conduct In-Stage Assessment

Responsibility: Project Manager and Independent Reviewer

Description: An In-Stage Assessment (ISA) is an independent review of the work products and deliverables developed or revised during each stage of the project lifecycle. The independent reviewer is typically a member of the Quality Assurance Team who is assigned to the software project and conducts all of the ISAs for the project.

An ISA does not require meetings with, or extra work by, the project team. All of the work products and deliverables needed for the review should be readily available in the Project File.

Schedule at least one ISA prior to the System Design Stage Exit process. Additional ISAs can be performed during the stage, as appropriate. The completion of System Design Document is an appropriate time to schedule an ISA.

Provide the reviewer with copies of all work products developed or revised during the System Design Stage including the Project Plan. The reviewer assesses the work products and deliverables to verify the following:

- The project is complying with the site's software engineering standards/best practices.
- Sound project management practices are being used.
- The project risks are identified and mitigated.

A description of the ISA process and the ISA report form are provided in the *In-Stage Assessment Process Guide.* A copy of the guide is provided in Appendix D.

Note: An ISA is an effective project management tool that is recommended for all projects regardless of size.

Work Product: An ISA report form is prepared by the independent reviewer and is used to identify open issues that need to be resolved in this stage. The report is delivered to the project manager and a copy should be placed in the Project File.

Activity: **6.12**
Conduct System Design Stage Exit

Responsibility: Project Manager

Description: The Stage Exit is a process for ensuring that projects are on target, within budget, on schedule, and meet the DOE and project standards identified in the Project Plan. The goal of a Stage Exit is to secure the approval of designated key individuals to continue with the project and to move forward into the next lifecycle stage.

Schedule the Stage Exit as the last activity of the System Design Stage. It is the responsibility of the project manager to notify the appropriate participants when a project is ready for the Stage Exit process and to schedule the Stage Exit meeting. All functional areas and the Quality Assurance representative involved with the project should receive copies of the work products and deliverables produced in this stage.

During the Stage Exit meeting, participants discuss open issues that will impact the Project Plan. The project manager should ensure that an acceptable action plan is developed for handling all open issues. At the conclusion of the meeting, concurrence is needed from the designated approvers to begin the next stage.

A description of the Stage Exit process is provided in the *Stage Exit Process Guide*. A copy of the guide is provided in Appendix E.

Note: A Stage Exit is an effective project management tool that is recommended for all software projects regardless of size. For small software projects, stages can be combined and addressed during one Stage Exit.

Work Product: A summary of the Stage Exit meeting is prepared by the project manager or a designee and distributed to the meeting attendees. The summary identifies any issues and action items needed to obtain concurrence prior to proceeding to the Programming Stage.