Software Engineering Methodology

Chapter 5.0
Functional Design Stage

Table of Contents

Chapter **Page**

Chapter: **5.0**
Functional Design Stage

Description: The functional design process maps the "what to do" of the Software Requirements Specification into the "how to do it" of the design specifications. During this stage, the overall structure of the software product is defined from a functional viewpoint. The functional design describes the logical system flow, data organization, system inputs and outputs, processing rules, and operational characteristics of the software product from the user's point of view. The functional design is not concerned with the software or hardware that will support the operation of the software product, or the physical organization of the data or the programs that will accept the input data, execute the processing rules, and produce the required output.

The focus is on the functions and structure of the components that comprise the software product. The goal of this stage is to define and document the functions of the software product to the extent necessary to obtain the system owner and users understanding and approval and to the level of detail necessary to build the system design.

Prototyping of system functions can be helpful in communicating the design specifications to the system owner and users. Prototypes can be used to simulate one function, a module, or the entire software product. Prototyping is also useful in the transition from the functional design to the system design.

Input: The following work products provide input to this stage.

- Project File
- Software Configuration Management Plan *(draft)*
- Continuity of Operations Statement/Plan
- Data Dictionary
- Requirements Traceability Matrix
- Software Requirements Specification
- Project Test Plan
- Acceptance Test Plan *(revised)*
- Design methodology
- Project Plan *(revised)*
- Software Quality Assurance Plan

High-Level

Activities: The remainder of this chapter is divided into sections that describe 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.

5.1 Determine Software Structure
5.2 Design Content of System Inputs and Outputs
5.3 Design User Interface
5.4 Design System Interfaces
5.5 Design System Security Controls
5.6 Build Logical Model
5.7 Build Data Model
5.8 Develop Functional Design
5.9 Initiate Procurement of Hardware and Software
5.10 Revise Project Plan
5.11 Conduct In-Stage Assessment
5.12 Conduct Functional Design Stage Exit

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

- Design records
- Logical model
- Data Dictionary
- Requirements Traceability Matrix *(expanded)*
- Functional Design Document
- Minutes from Functional Design Review
- Hardware and software procurement records
- Project Plan *(revised)*

A matrix showing the work products associated with each high-level activity is provided in *Exhibit 5.0-1, Functional 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 software products.

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 Preliminary Design Review will be conducted. This review is an important milestone in the design process. The time and resources needed to conduct the walkthroughs and Functional Design Review should be reflected 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 5.0-1. Functional Design Stage Activities and Work Products by Project Size

Work Activity		Project Size L	M	S	Work Product	Scheduled Deliverables L	M	S
5.1	Determine Software Structure	R	R	R	Design entities and dependencies	I	I	I
5.2	Design Content of System Inputs and Outputs	R	R	R	System input and output design	I	I	I
5.3	Design User Interface	R[2]	R[2]	R[2]	User interface design	I[2]	I[2]	I[2]
5.4	Design System Interfaces	R	R	R	System interface design	I	I	I
5.5	Design System Security Controls	R	R	R	System security control design	I	I	I
5.6	Build Logical Model	R	R	R	Logical model	R	R	R
5.7	Build Data Model	R	R	R	Data dictionary *(revised)*	R	R	R
5.8	Develop Functional Design	R	R	R	Requirements Traceability Matrix *(expanded)* Functional Design Document Functional Design Review minutes	R R R	R R R	R R R
5.9	Initiate Procurement of Hardware and Software	A	A	A	Procurement records	N	N	N
5.1	Revise Project Plan	R	R	A	Project Plan *(revised)*	R	R	A
5.11	Conduct In-Stage Assessment	R	R	A	ISA Report Form[1]	N	N	N
5.12	Conduct Functional Design Stage Exit	R	R	A	Stage Exit Meeting Summary	N	N	N

Size: L = Large, M = Medium, S = Small

Minimum Requirements: R = Required, A = As Appropriate, N = Not Appropriate

I = Input to other deliverables
[1] = Completed by reviewer
[2] = Can use existing plan/procedure

Bibliography: The following materials were used in the preparation of the Functional 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 4, Software Requirements Analysis. pp. 4-1 through 4-9.

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

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

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

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

9. 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.

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

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

13. U.S. Social Security Administration, Office of Systems, *Software Engineering Technology (SET) Manual*, Volume 1, 1990.

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

Activity: **5.1**
Determine Software Structure

Responsibility: Project Team Analysts

Description: A hierarchical approach is useful for determining the structure and components of the software product. Software system decomposition is one hierarchical approach that divides the software system into different levels of abstraction. Decomposition is an iterative process that continues until single purpose components (i.e., design entities or objects) can be identified. Decomposition is used to understand how the software product will be structured, and the purpose and function of each entity or object.

The goal of the decomposition is to create a highly cohesive, loosely coupled, and readily adapted design. A design exhibits a high degree of cohesion if each design entity in the program unit is essential for that unit to achieve its purpose. A loosely coupled design is composed of program units that are independent or almost independent.

Several reliable methods exist for performing system decomposition. Select a method that enables the design of simple, independent entities. Functional and object-oriented design are two common approaches to decomposition. These approaches are not mutually exclusive. Each may be applicable at different times in the design process.

The software system decomposition activity includes the following tasks.

5.1.1 Identify Design Entities
5.1.2 Identify Design Dependencies

Task: **5.1.1**
Identify Design Entities

Description: Design entities result from a decomposition of the software product requirements. A design entity is an element (or object) of a design that is structurally and functionally distinct from other elements and is separately named and referenced. The number and type of entities required to partition a design are dependent on a number of factors, such as the complexity of the software product, the design method used, and the programming environment. The objective of design entities is to divide the software product into separate components that can be coded, implemented, changed, and tested with minimal effect on other entities.

Attributes: A design entity attribute is a characteristic or property of a design entity. It provides a statement of fact about an entity. The following are common attributes that should be considered for each design entity.

- Assign a unique name to each entity.
- Classify each entity into a specific type. The type may describe the nature of the entity, such as a subprogram or module; or a class of entities dealing with a particular type of information.
- Describe the purpose or rationale for each entity. Include the specific functional and performance requirements for which the entity was created.
- Describe the function to be performed by each entity. Include the transformation applied to inputs by the entity to produce the desired output.
- Identify all of the external resources that are needed by an entity to perform its function.
- Specify the processing rules each entity will follow to achieve its function. Include the algorithm used by the entity to perform a specific task and contingency actions in case expected processing events do not occur.

Attributes, continued:

- Describe the data elements internal to each entity. Include information such as the method of representation, format, and the initial and acceptable values of internal data. This description may be provided in the data dictionary.

Work Product: Maintain a record of all design entities. The records will be integrated into the Functional Design Document. Place a copy of the design entity information in the Project File.

Review Process: Schedule structured walkthroughs to verify that the design entities are correct, complete, and possess the required attributes.

Task: **5.1.2**
Identify Design Dependencies

Description: Design dependencies describe the relationships or interactions between design entities at the module, process, and data levels. These interactions may involve the initiation, order of execution, data sharing, creation, duplication, use, storage, or destruction of entities.

Identify the dependent entities of the software system design, describe their coupling, and identify the resources required for the entities to perform their function. Also define the strategies for interactions among design entities and provide the information needed to perceive how, why, where, and at what level actions occur.

Dependency descriptions should provide an overall picture of how the software product will work. Data flow diagrams, structure charts, and transaction diagrams are useful for showing the relationship among design entities.

The dependency descriptions may be useful in producing the system integration plan by identifying the entities that are needed by other entities and that must be developed first. Dependency descriptions can also be used to aid in the production of integration test cases.

Work Product: Add specific dependency information to the design entity records. The records will be integrated into the Functional Design Document. Place a copy of the dependency information in the Project File.

Review Process: Schedule structured walkthroughs to verify that the design dependencies are correct and complete.

Activity: **5.2**
Design Content of System Inputs and Outputs

Responsibility: Project Team Analysts

Description: Design the content and format for each of the software product inputs and outputs based on the system input and output requirements identified during the Requirements Definition Stage. Involve the system owner and users in the design process to make certain that their needs and expectations are being met.

Procedure: Use the following procedure to implement the design process.

- Identify the types of electronic and printed input that will be accepted by the software product, such as data entered from source documents and files or records extracted from other systems.
- Identify the types of electronic and printed output that will be produced by the software product; such as data, records, or files; screen displays; and printed reports. Also identify the output that will be exported to other systems.
- Identify the specific input and output items that already exist and the items that will be created as part of the software product.
- Assign a name to each type of input and output and describe each item from a functional perspective.
- Identify the owner/originator of each type of input and output.
- Identify the frequency of each type of input and output.
- Design the content and format for each new input and output item or modify the format of existing items that must be changed to accommodate the new software product.

Work Product: Document the design for the system inputs and outputs in accordance with the project design standards. Discuss the designs with the system owner and users and submit completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the system input and output designs in the Project File.

Review Process: Schedule a structured walkthrough to verify that the system input and output designs are correct and complete.

Activity: **5.3**
Design User Interface

Responsibility: Project Team Analysts

Description: Design a user interface that is appropriate for the users, content, and operating environment for the software product. Determine interface levels for all categories of users. For interactive user environments, prototype the user interface. Arrange for users to experiment with the prototypes so that design weaknesses in the interface can be identified and resolved early. Use prototypes to gain user acceptance of the interface.

If the site or system owner's organization has an existing user interface standard, this standard should be used to specify the user interface for every software product developed for that organization. A user interface standard should be developed and maintained for each organization that does not have one.

Review the standard each time a new software product is planned to verify that the user interface is compatible with the software product's selected system architecture. For example, some DOS-based user interface standards would not be appropriate for a Windows-based software product.

The following tasks are involved in specifying the user interface.

5.3.1 Design Menu Hierarchy
5.3.2 Design Data Entry Screens
5.3.3 Design Display Screens
5.3.4 Design Online Help
5.3.5 Design System Messages

Basic Principles: The following basic principles can help improve the software product user interface when there is graphical, command-based, menu-driven, or block mode features.

- Give users control. Let them choose actions to perform.
- Give users feedback and progress reports. Tell them when the system is working and when an action is completed.
- Make sure programs, windows, and functions are consistent within and with other components of the software product.

Basic Principles,

continued:

- Be consistent in the format and wording of text.
- Keep it simple. White space is as important on the screen as on the printed page. Reduce screen clutter.
- Use special effects carefully and sparingly. Be sure color screens also work in one color--some users are colorblind, and some users have monochrome monitors. Use color consistently. Beeps and other sounds can be annoying; so let users turn sound off.
- Put information where it can be easily seen; avoid information in corners or borders.
- Limit the amount of information users must know. Offer choices instead of making users remember and enter information. Provide defaults, and make sure they are logical and satisfy a large number of users.
- Offer shortcuts. Keyboard shortcuts (e.g., hot keys) and command abbreviations help experienced users work more quickly.
- Help users get out of trouble. Provide messages that are understandable and that offer solutions.
- Let users reverse their actions. If an action will destroy something, identify the object of destruction and wait for a response.

Task: **5.3.1**
Design Menu Hierarchy

Description: Use the following guidelines to improve the design of menu hierarchies.

- Choose an organizing principle for the menu options, such as:
  - Expected frequency of use
  - Logical sequence of operations
  - Alphabetical order (should be used for horizontal word menus with five or more words)
- Put a meaningful title at the top of every menu.
- For full-screen menus, provide symmetric balance by centering the title and the menu options around the center axis of the screen.
- To facilitate scanning, put blank lines between logical groupings of menu options and after about every fifth option in a long list.
- Limit the number of menu choices to one screen.
- Use a menu option selection method that is consistent with the technology available at the user's workstation and the size of the software product being designed, such as:
  - Numbers
  - Letters or letter combinations
  - Cursor movement
- Provide a way for the user to leave the menu without performing any action. Be sure that the option to leave the menu describes the consequences of its selection.
- Words used for menu options should follow these rules:
  - Use words that clearly and specifically describe what the user is selecting.
  - Use common English words rather than computer or technical jargon. When space permits, spell out words completely.

Description, continued:

- Use simple, active verbs to tell users what actions will result from their choice. Try to start each option with a verb.
- Use parallel construction to describe the options.

- Minimize the highlighting used on a menu. Highlighting should be limited to situations where the user needs to know that there is an exception to the normal practice.
- Do not require the user to enter leading or trailing blanks or zeros, and do not include a default value on a menu.
- Display the menu options in mixed letters (i.e., upper and lower case).
- Organize menu hierarchies according to the tasks users will perform, rather than the structure of the software modules.

Work Product: Document the design for the menu hierarchy in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed design for their review and approval. The approved design will be incorporated into the Functional Design Document. Place a copy of the menu hierarchy design in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the menu hierarchy design is complete and logical.

Task: **5.3.2**
Design Data Entry Screens

Description: Use the following guidelines to improve the design of data entry screens.

- When the user must transcribe data directly from a source document to the screen, the layout of the screen should be similar to the layout of the source document.
- Group data fields into logical categories on the screen; provide a header that describes the contents of each category.
- Make areas of the screen that are not needed for data entry or commands inaccessible to the user.
- Do not require the user to enter information that is already available to the software or can be computed by it.
- Do not require the user to enter dimensional units, leading or trailing blanks, or zeros.
- Allow the user to enter data by character replacement.
- Put a caption describing the data to be entered adjacent to each data field; incorporate memory joggers into the caption.
- Justify data entries automatically.
- Display default values in data fields when appropriate.
- Provide context-sensitive help for data entry fields.

Work Product: Document the designs for the data entry screens in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the data entry screen designs in the Project File.

Review Process: Conduct a structured walkthrough to assure that the data entry screen designs are consistent, complete, and logical.

Task: **5.3.3**
Design Display Screens

Description: Use the following guidelines to design display screens that are easy to use and understand.

- Put a title on every display screen. The title should clearly and specifically describe the contents of the screen.
- Display only information that the user needs to know.
- Display data to the user in directly usable form.
- Provide symmetric balance to displays by centering titles and headings and by placing information on both sides of the center axis.
- Every display should indicate how to exit from the screen. Use consistent exit procedures.
- When the display continues over multiple screens, the screen should indicate where the user is in the display (e.g., Screen 1 of 3).
- Consider the skills of the users and the information they will manipulate when information is displayed in multiple windows.
- Data fields need to be grouped into logical categories or according to the structure of a source document (when there is one).
- Be consistent in the use of words and special characters.
- Display text conventionally in mixed letters (i.e., upper and lower case) and with appropriate punctuation. Avoid all uppercase letters. Put a blank line between paragraphs.
- Left justify text, and leave a ragged right margin.
- Avoid hyphenation of words between lines.
- Use abbreviations and acronyms only when they are significantly shorter than the full text and when they will be understood by the user.
- Be consistent with the format of information being displayed.

Table and List

Guidelines: Use the following guidelines to improve the design of online tables and lists.

- Put a meaningful label on the columns and, if appropriate, the rows of tables and lists. Continue the labels when a table or list extends over more than one screen.
- If data items are scrolled, the labels should be fixed on the screen and not be part of the scrolled area (they remain in place as the body of the table or list changes).
- If data items are continued on subsequent screens, the labels should be added to each screen.
- Arrange the items in a table or list in some recognizable order to facilitate scanning.
- Put items in a multiple column list in vertical columns that are read from left to right on the screen.
- Left justify columns of alphabetic data; right justify columns of numeric data or align them by the decimal point or other delimiter.
- Insert a blank line after about every fifth row in a long column.
- Insert a minimum of two spaces between the longest item in a column and the beginning of the next column.
- Start with a one (1) not a zero (0) when listed items are labeled by number.

Work Product: Document the design for the display screens in accordance with the project design standards. Discuss the designs with the system owner and users and submit the completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the display screen designs in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the display screen designs are consistent, complete, and logical.

Task: **5.3.4**
Design Online Help

Description: Online help is typically requested by users when they want to perform a new, complex, or infrequently used procedure, or when they do not know what else to do. The text of online help messages needs to be planned, drafted, and evaluated as carefully as print documentation. In addition, the layout and format of online help must be designed to deal with the special constraints imposed by the video screen.

Use online help to explain concepts, procedures, messages, menu choices, commands, words, function keys, and formats. Work with the users to identify the level of detail needed for online help. Determine whether the users need a one-line message at the bottom of the screen or a full online explanation with successive levels of detail. Effective online help messages tell users what the software product is doing, where they are in the sequence of screens, what options they have selected, and what options are available.

Guidelines: The following guidelines can improve the design of online help.

- Write online help messages in plain English.
  - Straightforward and reads as if it were spoken.
  - Clear, direct, and simple.
  - Effectively organized with a concern for what users need to know.
- Address the user directly as "you"; use the active voice.
- Use simple action verbs to describe procedures. Do not use nouns to replace pronouns, verbs, and adjectives.
- Describe procedures in logical order.
- Avoid computer terms or other jargon, such as:
  - Terms that are unique to the computer profession or to a particular company.
  - Terms that have a common meaning outside of the data processing environment, but a special meaning within it, such as *boot*, *abort*, *default*, and *utility*.

Guidelines, continued:

- Terms that are made up to describe some special function, such as *ungroup* and *dearchive*.

- Avoid humor in online documentation.
- Write in short complete sentences and paragraphs and use proper punctuation.
- Write sentences in the positive or simple negative. Avoid the passive voice and do not use double negatives.
- Use bullets, numbered lists, and tables to make it easier to find the most important information. Leave ample open space.
  - Use bulleted lists to explain options. Whenever a sentence lists options with commas between them, consider breaking up the text into a bulleted list.
  - Use numbered lists to show the steps in a process.
  - Use a table to explain two or more categories of information.
- Use examples to show users what they should enter and what the results will look like.
- Do not expect users to read more than about three screens of help at one time.
- Provide an orientation to the structure of the software product.
- Whenever possible display help text on the screen with the function or task that is being performed.
- Provide a direct route back to the function or task being performed.

Work Product: Document the design for online help in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed design for their review and approval. The approved design will be incorporated into the Functional Design Document. Place a copy of the online help design in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the online help design is consistent, complete, and logical.

Task: **5.3.5**
Design System Messages

Description: System messages are the various types of information that the system provides to the user such as status messages, user prompts, and error messages.

Status Messages: Status messages are important for giving users the feeling they are in control of the software. They tell users what the software is doing, where they are in the sequence of screens, what options they have selected, and what options are available.

User Prompts: Prompts inform the user to type data or commands or to make a simple choice.

- Use prompts to ask the user to make a simple choice or to enter data or commands. Be as specific as possible.
- Include memory aids in the prompt to help users type a response in the proper format and order, initiate infrequently used processes, or clearly identify exceptions to normal practice.
- When defaults are allowed with prompts, indicate clearly which default value will be initiated.

Error Messages: Error messages should allow users to recover from mistakes by making it clear what the mistake was and how to correct it. Error messages need to be specific about why a mistake was made.

- Design the software product to check for obvious errors.
- Be as specific as possible in describing the cause of an error. Do not use error codes.
- Do not assign blame to the user or the software in an error message. Use a neutral tone.
- Whenever possible, the error message should indicate what corrective action the user needs to take.
- Be consistent in the format, wording, and placement of messages.
- Consider describing error messages at more than one level of detail.

Work Product: Document the design for the system messages in accordance with the project design standards. Discuss the designs with the system owner and users and submit the completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the system message designs in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the system message designs are consistent, complete, and logical.

Activity: **5.4**
Design System Interfaces

Responsibility: Project Team Analysts

Description: Develop a design depicting how the software product will interface with other systems based on the system interface requirements identified in the Requirements Definition Stage. Submit the applicable interface designs for review by the system owner or system administrator for each system that will interface with the software product. Any incompatibilities with the interfaces will be identified early in the design process and corrective actions can be initiated to assure each interface is properly designed and coded.

Sample Issues: The following list provides some of the issues that should be considered when designing the system interfaces.

- System inputs and outputs
- Method of interface
- Volume and frequency of data
- Platform of interfacing system
- Format of data
- Automatic or manual initiation of interface
- Need for polling device(s)
- Verification of data exchange
- Validation of data

Work Product: Document the design(s) for the system interfaces in accordance with the project design standards. Discuss the designs with the system owner and users and submit completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the system interface designs in the Project File.

Review Process: Schedule a structured walkthrough to verify that the system interface designs are correct and complete.

Activity: **5.5**
Design System Security Controls

Responsibility: Project Team Analysts and Security Personnel

Description: Design the security controls that will be incorporated into the software product based on the security and access requirements identified during the Requirements Definition Stage. Design the security controls in conjunction with the site or system owner organization's Computer System Security Officer (CSSO) or the Assistant Computer Protection Program Manager (ACPPM).

Procedure: Use the following procedure to implement the design process.

- Identify the users and organizations that will have access to the software product. Indicate what access restrictions they will have. All persons in a work area may not have the same security access level. Measures should be taken to assure that sensitive materials and software requiring protection are not accessed by unauthorized individuals.

- Identify controls for the software product, such as the user identification code for system access and the network access code for the network on which the software product will reside.

- Identify whether access restrictions will be applied at the system, subsystem, transaction, record, or data element levels. Classified information must be protected in accordance with DOE directives.

- Identify physical safeguards required to protect hardware, software, or information from natural hazards and malicious acts.

- Identify communications security requirements.

Work Product: Document the design for the system security controls in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed design for their review and approval. The approved design will be incorporated into the Functional Design Document. Place a copy of the system security control design in the Project File.

Review Process: Schedule a structured walkthrough to verify that the system security controls are correct and complete. Include the CSSO or the ACPPM in the walkthrough.

References:

- DOE 5639.6, CLASSIFIED COMPUTER SECURITY PROGRAM, September 15, 1992.
- DOE HEADQUARTERS UNCLASSIFIED COMPUTER PROTECTION PLAN, February 1995.

Activity: **5.6**
Build Logical Model

Responsibility: Project Team Analysts

Description: The logical model defines the flow of data through the software system and determines a logically consistent structure for the software. Each module that defines a function is identified, interfaces between modules are established, and design constraints and limitations are described. The focus of the logical model is on the real-world problem or need to be solved by the software product.

A logical model has the following characteristics:

- Describes the final sources and destinations of data and control flows crossing the system boundary rather than intermediate handlers of the flows.
- Describes the net transfer of data across the system boundary rather than the details of the data transfer.
- Provides for data stores only when required by an externally imposed time delay.

When building a logical model, the organization of the model should follow the natural organization of the software product's subject matter. The names given to the components of the model should be specific. The connections among the components of the model should be as simple as possible.

Work Product: The logical model should be documented in user terminology and contain sufficient detail to obtain the system owner's and users' understanding and approval. Use data flow diagrams to show the levels of detail necessary to reach a clear, complete picture of the software product processes, data flow, and data stores.

Maintain the logical model and data flow diagrams for incorporation into the Functional Design Document. Place a copy of the logical model and data flow diagrams in the Project File. Keep the logical model and diagrams up-to-date. They will serve as a resource for planning enhancements during the Maintenance Stage, particularly for enhancements involving new functions.

Review Process: Schedule a structured walkthrough to verify that the logical model is correct, logical, and complete.

Activity: **5.7**
Build Data Model

Responsibility: Project Team Analysts

Description: A data model is a representation of a collection of data objects and the relationships among these objects. The data model is used to provide the following functions:

- Transform the business entities into data entities.
- Transform the business rules into data relationships.
- Resolve the many-to many relationships as intersecting data entities.
- Determine a unique identifier (keys) for each data entity.
- Add the attributes (facts) for each data entity.
- Document the integrity rules required in the model.
- Determine the data accesses (navigation) of the model.

Work Product: The data dictionary started in the Requirements Definition Stage is expanded in this stage to catalog every known data element used in the user's work and every system-generated data element. Data elements are documented in detail to include attributes, known constraints, input sources, output destinations, and known formats.

The data dictionary can serve as a central repository of information for both programmers and end users. The dictionary can include business rules, processing statistics, and cross-referencing information for multiple vendor environments.

To expand the data dictionary, define, analyze, and complete data definitions using the following steps.

- Identify data needs associated with various system features.
- Match (verify) data needs with the data dictionary.
- Match the data dictionary with specific data structures.

Work Product, continued:

- Create data record layouts.
- Ensure that all data can be maintained through add, change, or delete functions.

The data dictionary is further refined in the System Design Stage to complete the information on data elements, entities, files, physical characteristics, and data conversion requirements.

Sample Attributes:

The following is a sample of the type of attributes (information) that should be included for each element in a data dictionary.

Long data name (full name)
Short data name (abbreviation)
Alias
Data definition
Owner(s)
Occurrence(s)/key
Program mode
Input source(s); e.g., screens, external interfaces, system generated
Output destination(s); e.g., screens, reports, external interfaces
Values/meanings
Protection/security
Default value
Length/precision
Character set (type)
Format
Range
Surface edits
Remarks

Review Process:

Schedule a structured walkthrough to verify that the data dictionary is correct and complete.

Activity: **5.8**
Develop Functional Design

Responsibility: Project Team

Description: The software functional design describes how the software product will be structured to satisfy the requirements identified in the Software Requirements Specification. It is a description of the software structure, components, interfaces, and data necessary before coding can begin.

The software functional design is a model or representation of the software product that is used primarily for communicating software design information to facilitate analysis, planning, and coding decisions. It represents a partitioning of the software system into design entities and describes the important properties and relationships among those entities. Design descriptions may be produced as documents, graphic representations, formal design languages, records in a data base management system, and CASE tool dictionaries.

Within the functional design, the design entities can be organized and presented in several ways. The goal of this activity is to compile the design entities and their associated attributes in a manner that facilitates the access of design information from various viewpoints (e.g., project management, configuration management, quality assurance, and testing). Also, the design entities and their attributes must be described in terms that are understandable to the system owner and users.

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 functional design to the requirements. Place a copy of the expanded matrix in the Project File.

The following tasks are involved in developing the functional design.

5.8.1 Develop Functional Design Document
5.8.2 Conduct Functional Design Review

Task: **5.8.1**
Develop Functional Design Document

Description: The Functional Design Document defines the functions of the system in user terminology and provides a firm foundation for the development of the system design. The Functional Design Document should be written from the system owner/users' perspective. This document provides the owner/users with an opportunity to review and provide input to the software product design before system design work is completed.

Work Product: Prepare a draft Functional Design Document. Use the designs developed for inputs, outputs, user and system interfaces, and security controls as input to this document. Submit the draft document to the system owner and users for their review and approval. After making the changes needed to resolve problems found during the review, the approved Functional Design Document becomes an official agreement and authorization to use the functional design as the basis for developing the system design. Place a copy of the approved Functional Design Document in the Project File.

Review Process: Conduct structured walkthroughs as needed to assure that the Functional Design Document is accurate, complete, and describes the functional design in a manner that can be understood by the system owner and users.

The completion of the draft Functional 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: **5.8.2**
Conduct Functional Design Review

Description: The Functional Design Review is a formal technical review of the basic design approach. The primary goal of the Functional Design Review is to demonstrate the ability of the software design to satisfy the project requirements. The review should be a series of presentations by the project team to the system owner, users, functional area points-of-contact, and Quality Assurance representative. Vendors may be invited to participate in the Functional Design Review when an off-the-shelf software product or hardware item is being considered for the system architecture.

Conduct the Functional Design Review to perform the following verifications.

- Evaluate the progress, technical adequacy, and risk resolution of the selected design approach. Determine whether the approved design approach is being followed by the project team.
- Evaluate the progress, technical adequacy, and risk resolution of the selected test approach. Review the following items:
  - Organization and responsibilities of group conducting tests
  - Project Test Plan
  - Planned format, content, and distribution of test reports
  - Planned resolution of problems and errors identified during testing
  - Retest procedures
  - Change control and configuration management of test items
  - Special test tools not required as deliverables
- Evaluate the methodology to be used to meet quality assurance requirements.
- Establish the existence and compatibility of the physical and functional interfaces.

Description, continued:

- Determine whether the functional design embodies all of the software product requirements.
- Verify that the design represents software that can meet the functional, data, and interface requirements.
- Review the planned user interfaces to the software. Examples of the types of design information to review:
  - Operating modes for each display station. For each mode, the functions performed, the displays and controls used.
  - The format and content standards for each screen (e.g., data locations, spaces, abbreviations, the number of digits, all special symbols, alert mechanisms).
  - Control and data entry devices and formats (e.g., keyboards, special function keys, and cursor control).
  - The format of all data inputs and provisions for error detection and correction.
  - The format for all status and error messages and data printouts (e.g., formats, headings, data units, abbreviations, spacing, columns).
- Demonstrate any rapid design prototypes used to make design decisions.
- Identify potential high risk areas in the design and any requirements changes that could reduce risk.
- Review to assure that consideration has been given to optimizing the maintainability and maintenance aspects of the software product.

Review Items: The following items should be considered for review and evaluation during the Functional Design Review. Be prepared to discuss in technical detail any of these items within the scope of the review.

- Functional flows. Indicate how the computer software functional flows map the software and interface requirements to the individual high-level components of the software product.

Review Items,

continued:

- Storage allocation data. Describe the manner in which available storage is allocated to individual software components. Timing, sequencing requirements, and relevant equipment constraints used in determining the allocation should be included.

- Control functions. Describe the executive control and start/recovery features of the software product.

- Component structure. Describe the high-level structure of the software product, the reasons for choosing the components, the development methodology that will be used within the constraints of available computer resources, and any support programs that will be required in order to develop and maintain the software product and allocated data storage.

- Security. Identify the security requirements and provide a description of the techniques to be used for implementing and maintaining security within the software product.

- Computer software engineering facilities. Describe the availability, adequacy, and planned utilization of the computer software engineering facilities including both Government-provided and commercially available facilities.

- Computer software engineering facility versus the operational system. Describe any unique design features that exist in the functional design in order to allow use within the computer software engineering facility that will not exist in the operational software product. Provide information on the design of support programs not explicitly required for the operational system that will be generated to assist in the development of the software product.

- Development tools. Describe any special tools (e.g., simulation, data reduction, or utility tools) that are not deliverables, but are planned for use during software development.

- Test tools. Describe any special test systems, test data, data reduction tools, test computer software, or calibration and diagnostic software that are not deliverables, but are planned for use during software development.

Review Items, continued:

- Commercial resources. Describe commercially available computer resources, including any optional capabilities (e.g., special features, interface units, special instructions, controls, formats). Identify any limitations of commercially available equipment (e.g., failure to meet user interface, safety, and maintainability requirements) and identify any deficiencies.

- Existing documentation. Maintain a file and have available for review any existing documentation supporting the use of commercially available computer resources.

- Support resources. Describe the resources necessary to support the software product during engineering, installation, and operational state (e.g., operational and support hardware and software personnel, special skills, human factors, configuration management, testing support, documentation, and facilities/space management).

- Operation and support documentation. Describe the documentation that will be produced to support the operation and maintenance of the software product.

Work Product:

Create and distribute official meeting minutes for each 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 the system owner and users for review and notification of review performance as follows:

- Approval - indicates that the functional design is satisfactorily completed.

- Contingent Approval - indicates that the functional design is not considered accomplished until the satisfactory completion of resultant action items.

- Disapproval - indicates that the functional design is inadequate. Another Functional Design Review is required.

Activity: **5.9**
Initiate Procurement of Hardware and Software

Responsibility: Project Manager/Team

Description: Careful consideration should be given to purchasing off-the-shelf software before expending the time, resources, and costs associated with developing custom-built systems. Whenever possible, acquire off-the-shelf software to satisfy some or all of the project requirements. In addition, some projects may require the acquisition of hardware or software to support the design, code, and test processes.

Try to acquire a demonstration package of any proprietary software before completing the design specifications. The proprietary software may prove inadequate or inappropriate once it has been evaluated through hands-on use. Create a pilot of the software product to exercise the most important functions provided by the proprietary software as well as to obtain definite performance indications.

Initiate the procurement of any hardware or software well in advance of the planned need for these products. Adequate time must be allocated in the Project Plan timeline for the selection, procurement, installation, testing, and training associated with each vendor product.

The project team may assume all of the procurement, installation, and testing responsibilities, or the acquisition and testing of some hardware and software may be initiated by the functional area that is most familiar with the product. For example, a local area network engineering group may procure and test local area network or client/server software; a mainframe systems group may procure and test mainframe software.

Note: When the expected operating platform for a software product will require extensive procurement of hardware and software, it is recommended that procurement needs be addressed as early in the lifecycle as possible. If hardware and software acquisition requirements are known, develop the Acquisition and Installation Plans for all operating sites and initiate the procurement process. Review and, if necessary, revise the Acquisition and Installation Plans at the beginning of the Programming Stage. Requirements for the Acquisition and Installation Plans are provided in *Chapter 7, Programming Stage.*

Work Product: Place a copy of all software and hardware procurement records (e.g., justifications, approvals, purchase orders, and invoices) and the Acquisition and Installation Plans (if developed) in the Project File.

Activity: **5.10**
Revise Project Plan

Responsibility: Project Manager

Description: Once the Functional Design Review is completed and the functional design is baselined, determine if the project estimates for resources, cost, and schedule need to be revised and if the selected design approach is still appropriate for the size and complexity of the project.

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

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 assure that the Project Plan reflects the project's current status and adequately estimates the resources, costs, and schedule for the System Design Stage.

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

Activity: **5.11**
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 Functional Design Stage Exit process. Additional ISAs can be performed during the stage, as appropriate. The completion of the Functional Design Document is an appropriate time to schedule an ISA.

Provide the reviewer with copies of all work products developed or revised during the Functional 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.
- 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: **5.12**
Conduct Functional 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 Functional 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 System Design Stage.