Software Engineering Methodology

Chapter 4.0
Requirements Definition Stage

Table of Contents

Chapter **Page**

Chapter: **4.0**
Requirements Definition Stage

Description: The primary goal of this stage is to develop a basis of mutual understanding between the system owner/users and the project team about the requirements for the project. The result of this understanding is an approved Software Requirements Specification that becomes the initial baseline for software product design and a reference for determining whether the completed software product performs as the system owner requested and expected.

This stage involves development of a Software Configuration Management Plan to track and control work products, analysis of the system owner/users' business processes and needs, translation of those processes and needs into formal requirements, and planning the testing activities to validate the performance of the software product.

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

- Project File
- Description of user environment
- Statement of project scope and objectives
- Statement of high-level project requirements
- Functional area contact list and project profile
- Summary of platform options
- Statement of project feasibility
- Analysis of Benefits and Costs Report
- Feasibility Study Document
- Project Plan
- 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.

4.1 Develop Software Configuration Management Plan
4.2 Select Requirements Analysis Methodology
4.3 Define Project Requirements
4.4 Compile Project Requirements

High-Level Activities, continued:

4.5 Establish Functional Baseline
4.6 Develop Project Test Plan
4.7 Develop Acceptance Test Plan
4.8 Select Design Methodology
4.9 Revise Project Plan
4.10 Conduct In-Stage Assessment
4.11 Conduct Requirements Definition 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 content 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.

- Software Configuration Management Plan
- Description of analysis methodology
- Records of all project requirements
- User-oriented requirements manual *(optional)*
- Continuity of Operations Statement/Plan
- Data Dictionary
- Requirements Traceability Matrix
- Software Requirements Specification
- Project Test Plan
- Acceptance Test Plan *(draft)*
- Design methodology
- Project Plan *(revised)*

A matrix showing the work products associated with each high-level activity is provided in *Exhibit 4.0-1, Requirements Definition 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 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. The time and resources needed to conduct the walkthroughs should be reflected in the project resources, schedule, and work breakdown structure.

Exhibit 4.0-1. Requirements Definition Stage Activities and Work Products by Project Size

Work Activity		Project Size L	M	S	Work Product	Scheduled Deliverables L	M	S
4.1	Develop Software Configuration Management Plan	R	R	R[2]	Software Configuration Management Plan	R	R	R[2]
4.2	Select Requirements Analysis Methodology	R	R	R	Description of analysis methodology	I	I	I
4.3	Define Project Requirements	R	R	R	Records of project requirements User-oriented requirements manual *(optional)* Continuity of Operations Statement/Plan Data Dictionary	I O R R	I O R R	I O R R
4.4	Compile Project Requirements	R	R	R	Requirements Traceability Matrix Software Requirements Specification *(draft)*	R R	R R	A R
4.5	Establish Functional Baseline	R	R	R	Software Requirements Specification *(final)*	R	R	R
4.6	Develop Project Test Plan	R	R	A	Project Test Plan	R	R	A
4.7	Develop Acceptance Test Plan	R	R	R	Acceptance Test Plan *(draft)*	R	R	R
4.8	Select Design Methodology	R[2]	R[2]	R[2]	Design Methodology	R[2]	R[2]	R[2]
4.9	Revise Project Plan	R	R	A	Project Plan *(revised)*	R	R	A
4.1	Conduct In-Stage Assessment	R	R	A	ISA Report Form[1]	N	N	N
4.11	Conduct Requirements Definition 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 Applicable

I = Input to other deliverables
O = Optional work product
[1] = Completed by reviewer
[2] = Can use existing plan

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

Bibliography: The following materials were used in the preparation of the Requirements Definition Stage chapter.

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

2. Fairley, Richard E., *Software Engineering Concepts,* McGraw-Hill Book Company, New York.

3. *Software Engineering Handbook,* Chapter 4, Software Requirements Analysis.

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

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

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

7. 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.

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

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

Activity: **4.1**
Develop Software Configuration Management Plan

Responsibility: Project Manager

Description: Configuration management is a set of procedures used to control changes to the project during all stages of the software lifecycle. A Software Configuration Management Plan that describes the configuration management procedures is required for each software project. Based on the complexity of the project and the anticipated volume of changes, a plan can be developed for a specific project, an existing plan can be modified to suit the requirements of a project, or a plan can be developed to manage all of the projects supporting a particular system owner's organization.

The plan is developed early in the lifecycle to ensure the control of changes as soon as the project requirements are approved and baselined. In this stage, the plan addresses activities that are platform independent, such as identifying the items that will be placed under configuration management. As the project progresses through the lifecycle stages, the plan is expanded to reflect platform specific activities.

Work Product: Prepare a Software Configuration Management Plan document if an existing plan is not available or applicable for the project. Place a copy of the Software Configuration Management Plan in the Project File.

Review Process: Conduct a structured walkthrough to validate that the configuration management approach, the configuration identification, change control, status accounting, and auditing procedures are appropriate for the project.

Activity: **4.2**
Select Requirements Analysis Methodology

Responsibility: Project Manager/Team

Description: A requirements analysis methodology is the set of data collection and analysis techniques (e.g., user interviews and rapid prototyping) combined with the lifecycle requirements standards (e.g., tracing the requirements through all lifecycle activities) that are used to identify the project requirements and to define exactly what the software product must do to meet the system owner/users' needs and expectations. When appropriate, the methodology must include techniques for collecting data about users at more than one geographic location and with different levels and types of needs.

The requirements analysis methodology should be in harmony with the type, size, and scope of the project; the number, location, and technical expertise of the users; and the anticipated level of involvement of the users in the data collection and analysis processes. The methodology should ensure that the functionality, performance expectations, and constraints of the project are accurately identified from the system owner/users' perspective. The methodology should facilitate the analysis of requirements for their potential impact on existing operations and business practices, future maintenance activities, and the ability to support the system owner's long-range information resource management plans.

It is advantageous to select a methodology that can be repeated for similar projects. This allows the project team and the system owner/users to become familiar and comfortable with the methodology. Discuss the analysis methodology with the system owner and users to make sure they understand the process being used, their role and responsibilities in the process, and the expected format of the output (e.g., how the requirements will be organized and described).

Work Product: Create a description of the analysis methodology and share it with all members of the project team, system owner, and users. Place a copy of the analysis methodology description in the Project File.

Review Process: Conduct a structured walkthrough to verify that the requirements analysis methodology is appropriate for the scope and objectives of the project. A structured walkthrough is not needed when the methodology has been used successfully on similar projects for the same system owner/user environment.

Activity: **4.3**
Define Project Requirements

Responsibility: Project Manager/Team

Description: Use the project scope, objectives, and high-level requirements as the basis for defining the project requirements. The questions used to define the project objectives may be helpful in developing the project requirements. The goals for defining project requirements are to identify what functions are to be performed on what data, to produce what results, at what location, and for whom. The requirements must focus on the software products that are needed and the functions that are to be performed. Avoid incorporating design issues and specifications in the requirements.

Requirements should be specified as completely and thoroughly as possible. The requirements must support the system owner's business needs, information resource management long-range plans, and the organizational and Departmental missions. When requirements are being defined, it is not sufficient to state only the requirements for the problems that will be solved; all of the requirements for the project must be captured.

Attributes: Each requirement must be stated as a unique objective with the following attributes. The existence of these attributes must be verified prior to the delivery of the Software Requirements Specification later in the Requirements Definition Stage.

- Necessary - Absolute requirements that are to be verified are indicated by "must" or "shall". Goals or intended functionality are indicated by "will".
- Correct - Each requirement is an accurate description of a feature or process of the software product.
- Unambiguous - The statement of each requirement denotes only one interpretation.
- Complete - Each requirement describes one result that must be achieved by the software product. The requirement should not describe the means of obtaining the result.
- Consistent - Individual requirements are not in conflict with other requirements.

Attributes, continued:

- Verifiable (testable) - Each requirement is stated in concrete terms and measurable quantities. A process should exist to validate that the software product (when developed) will satisfy the set of requirements.
- Modifiable - The structure and style of the requirements are such that any necessary changes to the requirements can be made easily, completely, and consistently.
- Traceable - The origin of each requirement is clear and can be tracked in future development activities and tests.

Identification System: The creation of a standard identification system for all requirements is required in order to facilitate configuration control, requirements traceability, and testing activities. The identification system must provide a unique designator for each requirement. For example, the identification system can classify the requirements by type (e.g., functional, input, or computer security). Within each type classification, the requirements can be assigned a sequential number. Select an identification system that is appropriate for the scope of the project.

Changes: As the project evolves, the requirements may change or expand to reflect modifications in the users' business plans, design considerations and constraints, advances in technology, and increased insight into user business processes. A formal change control process must be used to identify, control, track, and report proposed and approved changes. Approved changes in the requirements must be incorporated into the Software Requirements Specification in such a way as to provide an accurate and complete audit trail of the changes. This change control process should be an integral part of the project's Software Configuration Management Plan.

Tasks: The following tasks are involved in developing project requirements.

4.3.1	Define Functional Requirements
4.3.2	Define Input and Output Requirements
4.3.3	Define Performance Requirements
4.3.4	Define User Interface Requirements
4.3.5	Define System Interface Requirements
4.3.6	Define Communication Requirements
4.3.7	Define Computer Security and Access Requirements
4.3.8	Define Backup and Recovery Requirements
4.3.9	Define Data Requirements
4.3.10	Define Implementation Requirements

Task: **4.3.1**
Define Functional Requirements

Description: Functional requirements define what the software product must do to support the system owner's business functions and objectives. The functional requirements should answer the following questions.

- How are inputs transformed into outputs?
- Who initiates and receives specific information?
- What information must be available for each function to be performed?

Identify requirements for all functions whether they are to be automated or manual. Describe the automated and manual inputs, processing, outputs, and conditions for all functions. Include a description of the standard data tables and data or records that will be shared with other applications. Identify the forms, reports, source documents, and inputs/outputs that the software product will process or produce to help define the functional requirements.

A functional model should be developed to depict each process that needs to be included. The goal of the functional model is to represent a complete top-down picture of the software product.

Flow diagrams should be used to provide a hierarchical and sequential view of the system owner's business functions and the flow of information through the processes.

Work Product: Maintain a record of all functional requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the functional requirements in the Project File.

Sample Functional Requirement: *The selection criteria for the extraction of records shall be the occurrence of the letters "EW" in the Budget and Reporting Code field.*

Optional Work Product: Consider developing an optional work product that defines how the final software product will operate to support the system owner organization's business functions and objectives. This user-oriented requirements manual would identify processes in a narrative form from the user's perspective and would include requirements for all functions whether they are to be automated or manual. A functional description can be developed to depict each process

Optional Work Product, continued:

that will be provided. The goal is to present a complete top-down picture of the software product. This user-oriented requirements manual can be used as an aid in validating the functional requirements and serves as the basis for the user documentation. If a test group outside the project team is used, the test group can work with the project team to develop the manual.

Review Process:

Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the functional requirements.

Task: **4.3.2**
Define Input and Output Requirements

Description: Describe all manual and automated input requirements for the software product such as data entry from source documents and data extracts from other applications.

Describe all output requirements for the software product such as printed reports, display screens, and files.

Work Product: Maintain a record of all input and output requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the input and output requirements in the Project File.

Sample Input Requirement: *The application must automatically assign a unique, sequential Employee Number to each employee record that is entered into the data base.*

Sample Output Requirement: *All reports that contain Privacy Act data must include a warning statement in the report header information.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the input and output requirements.

Task: **4.3.3**
Define Performance Requirements

Description: Performance requirements define how the software product must function (e.g., hours of operation, response times, and throughput under detailed load conditions). The information gathered in defining the project objectives can translate into very specific performance requirements; (e.g., if work performed for an organization is mission essential to the Department, the hours of operation and throughput will be critical to meeting the mission). Also, Government and DOE policy can dictate specific availability and response times.

Work Product: Maintain a record of all performance requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the performance requirements in the Project File.

Sample Performance Requirement: *The application must be available for use from 8:00 a.m. to 5:00 p.m. Monday through Friday.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the performance requirements.

Task: **4.3.4**
Define User Interface Requirements

Description: The user interface requirements should describe how the user will access and interact with the software product, and how information will flow between the user and the software product.

Interface Issues: The following are some of the issues that should be considered when trying to identify user interface requirements.

- The users' requirements for screen elements, navigation, and help information.
- The standards for the programmatic organization, DOE, Government, and industry that apply to user interfaces.
- The range of skill levels of the users who will access and use the software product.
- The range of work that the users will be performing with the software product.

Define the user interface requirements by identifying and understanding what is most important to the user, not what is most convenient for the project team. Work with the system owner and users to develop a set of user interface requirements that can be used for all automated products for the system owner's organization. A standard set of user interface requirements will simplify the design and code processes, and ensure that all automated products have a similar look and feel to the users. When other constraints (such as a required interface with another application) do not permit the use of existing user interface standards, an attempt should be made to keep the user interface requirements as close as possible to the existing standard.

Work Product: Maintain a record of all user interface requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the user interface requirements in the Project File.

Sample User Interface Requirement: *All data entry screens must include a unique screen identification number.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the user interface requirements.

Task: **4.3.5**
Define System Interface Requirements

Description: The hardware and software interface requirements must specify hardware and software interfaces required to support the development, operation, and maintenance of the software product. The following information should be considered when defining the hardware and software interface requirements.

- System owner's and users' computing environment.
- Existing or planned software that will provide data to or accept data from the software product.
- Other organizations or users having access to the software product.
- Purpose or mission of interfacing software.
- Common users, data elements, reports, and sources for forms/events/outputs.
- Timing considerations that will influence sharing of data, direction of data exchange, and security constraints.
- Development constraints such as the operating system, data base management system, language compiler, tools, utilities, and network protocol drivers.
- Standardized system architecture defined by hardware and software configurations for organizations, programmatic offices, or telecommunications programs.

Work Product: Maintain a record of all system interface requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the system interface requirements in the Project File.

Sample System Interface Requirement: *The application must extract records with the following position status indicators from the HRIS mainframe application: EN, X, D, P, T or NW.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the system interface requirements.

Task: **4.3.6**
Define Communication Requirements

Description: The communication requirements define connectivity and access requirements within and between user locations and between other groups and applications.

The following factors should be considered when defining communication requirements.

- Communication needs of the user and customer organizations.
- User organization's existing and planned telecommunications environment (e.g., LANs, WANs, and dial-up).
- Projected changes to the current communication architecture, such as the connection of additional local and remote sites.
- Limitations placed on communications by existing hardware and software including:
  - existing user systems
  - existing applications that will interface with the software product
  - existing organizations that will interface with the software product
- Organization, Government, and industry standards that define communication requirements and limitations

Work Product: Maintain a record of all communication requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the communication requirements in the Project File.

Sample Communication Requirement: *The application must execute online in the organization's local area network environment.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability accuracy, and completeness of the communications requirements.

Task: **4.3.7**
Define Computer Security and Access Requirements

Description: Develop the computer security requirements in conjunction with the system owner's Computer System Security Officer (CSSO) or the Assistant Computer Protection Program Manager (ACPPM). This involvement affords early determination of classifications and levels of protection required for the software product.

If a software product under development processes sensitive personal information, appropriate safeguards must be established to protect the information from accidental disclosure.

Implement applicable security procedures to assure data integrity and protection from unauthorized disclosure, particularly during development efforts. The organization that owns the data defines the data classification. The project team must be aware of all the types of data and of any classified or proprietary algorithms used in the software product.

Procedure: Use the following procedure to determine computer security requirements.

1. Identify the types of data that will be processed by the software product.

2. Determine preliminary data protection requirements.

    a. For software products processing classified information refer to DOE 5639.6, CLASSIFIED COMPUTER SECURITY PROGRAM, September 15, 1992, attachment III, page III-19, paragraph 5.c.(4) - Applications Software.

    b. For software products processing unclassified information, refer to DOE HEADQUARTERS UNCLASSIFIED COMPUTER PROTECTION PLAN, dated December 1993.

    c. For software products processing sensitive information refer to Chapter 5 of the DOE HEADQUARTERS UNCLASSIFIED COMPUTER PROTECTION PLAN.

    d. For software products processing sensitive personal information, contact the Freedom of Information Office for coordination and assistance in complying with DOE 1800.1A, PRIVACY ACT.

Procedure, continued:

e. For software products that are considered to be mission essential refer to paragraph 5.4.1.3 and Chapter 8 of DOE HEADQUARTERS UNCLASSIFIED COMPUTER PROTECTION PLAN.

3. Coordinate with the owner of the host platform to identify existing supporting computer security controls, if applicable.

4. Incorporate security requirements into the Software Requirements Specification.

Work Product: Maintain a record of all security and access requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the security and access requirements in the Project File.

Sample Access Control Questions: The following list provides sample questions that can be used to help define the access controls for the software product.

- What access restrictions are placed on the users by their organization or programmatic office?
- What are the audit and other checking needs for the software product?
- What separation of duties, supervisory functions related to control, operating environment requirements, or other functions will impact the software product?
- What measures will be used to monitor and maintain the integrity of the software product and the data from the user's viewpoint?

Sample Security Requirement: *The application must maintain a record of all user access attempts sorted by authorized and unauthorized users.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the computer security and access requirements.

References:

- DOE Headquarters Computer Protection Plan (CPP) for Unclassified Systems, February 1995, describes the unclassified computer security program for Headquarters organizations.
- DOE Order 1360.2B, UNCLASSIFIED COMPUTER SECURITY PROGRAM, provides guidance for organizations to implement a computer security program for sensitive information.
- DOE Order 5639.6, CLASSIFIED COMPUTER SECURITY PROGRAM, provides guidance for classified systems.
- DOE 1800.1, PRIVACY ACT

Task: **4.3.8**
Define Backup and Recovery Requirements

Description: Develop the requirements for data backup, recovery, and operation startup for the software product in conjunction with the site authority for continuity of operations. If a software product has been defined as mission essential, a Continuity of Operations Plan must be developed. A checklist is provided in *Exhibit 4.3-1, Checklist for Identifying Mission-Essential Software*, to determine if the software is mission essential.

Work Product: If a software product is determined to be mission essential, a Continuity of Operations Plan must be developed. If the software product is not mission essential, a continuity of operations statement is required. Two samples of continuity of operations statements that are appropriate for software that is not mission essential are provided after the checklist. Place a copy of the Continuity of Operations Statement or Plan in the Project File.

Review Process: Conduct structured walkthroughs as needed to assure the necessity, testability, accuracy, and completeness of the backup and recovery requirements.

Reference: *Disaster Recovery Program Guidelines;* Department of Energy; Office of Information Resource Management; Policy, Plans, and Oversight, July 1991.

Exhibit 4.3-1. Checklist for Identifying Mission-Essential Software

The checklist is intended to be used to help identify software products that are mission essential. If a "yes" answer is selected for one or more of the criteria, the software product is mission essential and a Continuity of Operations Plan must be developed.

	Criterion	Yes	No
1	Inability to perform function adversely affects national security.		
2	Inability to perform function adversely affects safety of individuals.		
3	Needed for military effort and civil defense activities during a national emergency.		
4	Needed for mobilization and protection of material and manpower during national emergency.		
5	Function required for maintenance of public health, safety, and order.		
6	Maintains records essential to preservation of legal rights.		
7	Large financial loss incurred with inability to perform functions.		
8	Large expense incurred if performing function by other means.		
9	Primary repository of information reported to Congress or other agencies.		
10	Critical for compliance with federal regulatory requirements.		
11	Sole source of data unobtainable by other means, or not easily recreated.		

Mainframe Sample Statement:

The backup and recovery of the Human Resources Information System (HRIS) comes under the umbrella of the Virtual Machine system backup and recovery procedure. The Virtual Machine system is backed up daily by VMBACKUP under the control of VMSCHEDULE, an automatic job scheduler software package. It is possible to reconstruct HRIS to its state just prior to any system crash by restoring the data base using VMBLIST, another software package. Additionally, HRIS has a software utility called CALLADM, which, with the appropriate parameters, can provide backup and restore capabilities at the directory record level. This utility is fully documented in Appendix F of the HRIS Installation and Reference Guide. The Virtual Machine system is backed up once a week and the tapes are vaulted offsite.

Microcomputer Sample Statement:

The Human Resources Information System (HRIS) is backed up daily by the HRIS data base administrator, using HRIS system utilities. Both the data base and the HRIS log files are backed up. If there is a media failure, it is possible to reconstruct the data base to its state as of the most recent backup of the log files, using an automated procedure. It is possible to perform backups while HRIS users are connected and transactions are in progress, as well as when they are not. HRIS has a Continuity of Operations Plan that allows its users to continue operations using a server in Germantown, Maryland should the local area network (LAN) become unavailable for a significant period. HRIS will be included in the Continuity of Operations Plan being developed for the Human Resources and Administration LAN. This Continuity of Operations Plan is scheduled for completion during the current fiscal year.

Task: **4.3.9**
Define Data Requirements

Description: Data requirements identify the data elements and logical data groupings that will be stored and processed by the software product. The identification and grouping of data begins during the Requirements Definition Stage and is expanded in subsequent stages as more information about the data is known.

Work Product: The major output of the data requirements identification process is a data dictionary. A data dictionary provides an ordered set of definitions about data inputs and outputs, and data stores. In the Requirements Definition Stage, the data dictionary contains a minimum amount of information about data elements such as definitions of the entities, how the data are stored, and data flows to or from other applications. The data dictionary is refined during the design stages as data elements are documented in more detail, and the logical groupings of data elements are formed into interrelated tables or record descriptions.

Work Product: Maintain a record of all data requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the data requirements in the Project File.

Sample Data Requirement: *Records imported from HRIS and ABCD must be matched to the application using the employee's social security number.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the data requirements.

Task: **4.3.10**
Define Implementation Requirements

Description: Describe the requirements anticipated for implementing the software product (e.g., user production cycle). The high-level implementation requirements are identified early in the lifecycle to support decisions that need to be made for the software engineering approach. The implementation requirements are expanded into a full implementation approach during the design stages.

The following paragraphs provide highlights of some of the implementation requirements that need to be considered.

Operating Environment: Identify any capacity restrictions on the existing hardware or software that needs to be addressed and identify any hardware or software that needs to be acquired (e.g., communication hardware, file servers, off-the-shelf software, network interface cards, and LAN utilities). If hardware or software must be acquired, identify the necessary acquisition activities. These activities include preparing specifications, estimating costs, scheduling procurement activities, selection, installation, and testing.

Conversion: Identify requirements for converting data from an existing or external application to the new software product. Consider requirements for data entry, data protection, computer time, conversion programs, personnel, and other resources that will be needed. Also identify the requirements for the conversion of software, if necessary. Implementing a new application may involve converting software from one environment to another, or modifying software to interface with other applications. Include requirements for testing the conversion process and validating that it was successfully accomplished.

Installation: Identify the installation requirements for any new hardware, operating system, or software. For hardware installations, consider environmental factors such as air conditioning, power supply, and security requirements. For software installations, consider proprietary software such as data base management systems. For application software, consider the installation of the application's programs, parallel operation of the old and new applications, or the cutover from a test to a production environment. Hardware and software installation must be coordinated with the work cycles of the user organization to create a minimum of disruption, and to assure that data are available as needed. Installation must be scheduled to assure that, when data conversion is necessary, the needed data are protected.

Training: Identify the specific training needs for various categories of users and administrators. Also identify training requirements for personnel time, computer time, training facilities, and training data base(s).

Documentation: Identify requirements for the development and distribution of operational documentation for software support personnel and user documentation. Operational documentation may include job control procedures and listings, operational instructions, system administration responsibilities, archiving procedures, and error recovery. User documentation includes the users manual, step-by-step instructions, online documentation, and online help facilities.

Work Product: Maintain a record of all implementation requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the implementation requirements in the Project File. This information will also be used to develop an Implementation Plan in the Functional Design Stage.

Sample Conversion Requirement: *All Julian dates found in the extract files must be converted to Gregorian dates.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the implementation requirements.

Activity: **4.4**
Compile Project Requirements

Responsibility: Project Manager/Team

Description: Compile the requirements gathered during the requirements analysis process in preparation for the development and delivery of the draft Software Requirements Specification. The following steps should be performed as part of the requirements compilation activity.

- Select and use a standard format for describing the requirements.
- Present the logical and physical requirements without dictating a physical design or technical solutions.
- Write the requirements in nontechnical language that can be fully understood by the system owner and users.
- Organize the requirements into meaningful groupings (e.g., all security-related requirements or all requirements for generating reports).
- Develop a numbering scheme for the unique identification of each requirement.
- Select a method for: (1) tracing the requirements back to the sources of information used in deriving the requirements (e.g., specific system owner/user project objectives); and (2) threading requirements through all subsequent lifecycle activities (e.g., testing).

Tasks: The following tasks are involved in the compilation of the project requirements.

4.4.1 Develop Requirements Traceability Matrix
4.4.2 Develop Software Requirements Specification

Task: **4.4.1**
Develop Requirements Traceability Matrix

Description: A requirements traceability matrix is a table used to trace project lifecycle activities and work products to the project requirements.

Every project requirement must be traceable back to a specific project objective(s) described in the Project Plan. This traceability assures that the product will meet all of the project objectives and will not include inappropriate or extraneous functionality.

All work products developed during the design, code, and testing processes in subsequent lifecycle stages must be traced back to the project requirements described in the Software Requirements Specification. This traceability assures that the product will satisfy all of the requirements and remain within the project scope.

It is also important to know the source of each requirement, so that the requirements can be verified as necessary, accurate, and complete. Meeting conference records, user survey responses, and business documents are typical sources for project requirements.

Work Product: Develop a matrix to trace the requirements back to the project objectives identified in the Project Plan and forward through the remainder of the project lifecycle stages. Place a copy of the matrix in the Project File. Expand the matrix in each stage to show traceability of work products to the requirements and vice versa.

Sample Traceability Matrix: One method for tracing requirements is a threading matrix that groups requirements by project objectives. Under each project objective, the source of the requirement, the unique requirement identification number, and the lifecycle activities are listed in columns along the top and the project requirements in rows along the left side. As the project progresses through the lifecycle stages, a reference to each requirement is entered in the cell corresponding to the appropriate lifecycle activity. *Exhibit 4.4-1, Sample Requirements Traceability Matrix,* provides a sample matrix format.

Exhibit 4.4-1. Sample Requirements Traceability Matrix

Requirement	Source of Requirement	Unique Number	Design Spec.	Program Module	Test Spec.
Objective 1: Security					
The software product shall have four user access levels with the capability to add new access levels in the future.	conference record dated 5/19/95	SYSADM 1.0			
Each user access level shall have a unique designation.	conference record dated 5/19/95	SYSADM 1.1			
One user access level shall allow read-only access to the production data base.	conference record dated 5/19/95	SYSADM 1.2			
The second user access level shall allow read and write access to the production data base.	conference record dated 5/19/95	SYSADM 1.3			
The third user access level shall allow read, write, and delete access to the production data base and read-only access to the history data base.	conference record dated 5/19/95	SYSADM 1.4			
The fourth user access level shall allow read, write, and delete access to all application data bases.	conference record dated 5/19/95	SYSADM 1.5			

Task: **4.4.2**
Develop Software Requirements Specification

Description: The Software Requirements Specification describes the inputs to be supplied by the user or other sources, the processing that needs to occur, and the outputs desired by the user or required by interfacing systems. The emphasis should be placed on specifying product functions without implying how the product will provide those functions. This approach provides maximum flexibility for the product designers. The how-to of product implementation is determined in the design stages.

Work Product: Prepare the Software Requirements Specification by integrating all of the requirements developed during this stage. Several formats are available for organizing the requirements information (e.g., from a functional perspective or a data processing perspective).

Document all design constraints including processing, performance, interface, resource, safety, security and reliability requirements. Define data constraints such as limits, formats, messages, commands, and displays.

Review Process: Conduct structured walkthroughs as needed to ensure that the Software Requirements Specification is accurate, complete, and expresses the requirements in a manner that can be understood by the system owner.

The completion of the draft Software Requirements Specification 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.

Activity: **4.5**
Establish Functional Baseline

Responsibility: Project Manager/Team

Description: The functional baseline, sometimes called a system requirements baseline, is the main technical work product of the Requirements Definition Stage. The system requirements are baselined after the system owner's formal approval of the Software Requirements Specification. Once the requirements are baselined, any changes to the requirements must be managed under change control procedures established in the Software Configuration Management Plan. Approved changes must be incorporated into the Software Requirements Specification.

Work Product: Prepare the final Software Requirements Specification and submit to the system owner and users for their review and approval. The approved Software Requirements Specification is the official agreement and authorization to use the requirements for the software product design. Approval implies that the requirements are understood, complete, accurate, and ready to be used as the basis for the subsequent lifecycle stages.

It is important for the system owner/users to understand that changes to the approved Software Requirements Specification affect the project scope and therefore can change the project cost, resources, or schedule. It is the responsibility of the project manager and project 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.

Place a copy of the approved Software Requirements Specification in the Project File.

Review Process: The Software Requirements Specification should be reviewed by the system owner and users. After making the changes needed to resolve problems found during the review, the functional baseline is formally established upon receipt of the system owner's approval.

Activity: **4.6**
Develop Project Test Plan

Responsibility: Project Manager

Description: The Project Test Plan is a narrative and tabular description of the test activities planned for the project. The Project Test Plan should establish the testing necessary to validate that the project requirements have been met and that the deliverables are at an acceptable level in accordance with existing standards. The plan also ensures that a systematic approach to testing is established and that the testing is adequate to verify the functionality of the software product.

The Project Test Plan includes the resources, project team responsibilities, and management techniques needed to plan, develop, and implement the testing activities that will occur throughout the lifecycle. If individuals outside of the project team perform system and acceptance testing, the plan includes the responsibilities and relationships of external test groups.

In this stage, the plan is written at a high level and focuses on identifying test methodologies and test phases. Detailed information about test products (i.e., test plans, test procedures, and test reports) is added to the Project Test Plan as the project progresses through subsequent lifecycle stages.

Development of the Project Test Plan is the responsibility of the project manager. If a test group outside the project team will be involved in any test phase, the project manager must coordinate the Project Test Plan with each test group.

The Project Test Plan must be reviewed and approved by the system owner prior to conducting any tests.

Preparation of the Project Test Plan involves the following tasks.

4.6.1 Identify Test Methodologies
4.6.2 Identify Test Phases
4.6.3 Identify Test Environment Requirements

Note: For small software projects, a formal Project Test Plan may not be necessary; however a test approach and testing are required.

Work Product: When the Project Test Plan is complete, it should contain the following information:

- Describe the occurrence and timing of the test phases in the lifecycle and the entrance and exit criteria for each test phase.
- Specify the test products at each test phase. Describe the types and scope of the testing activities to be performed on each component of the application and the group who is responsible to produce them.
- Map what requirements are verified in what test phase.
- Establish the criteria for evaluating the test results of each test phase.
- Make an initial determination of the resources necessary to accomplish the testing.
- Identify the appropriate person or group to conduct each type of testing activity.
- Outline the test environment (hardware, software, test tools, and data) needed to conduct the tests.
- Develop a preliminary schedule for executing the test activities.

Place a copy of the Project Test Plan in the Project File.

Review Process: Conduct structured walkthroughs to assure the Project Test Plan document adequately describes all testing activities, test schedules, test products, test responsibilities, the testing methodology, and the required resources.

Task: **4.6.1**
Identify Test Methodologies

Description: The Project Test Plan should specify the testing methodologies planned for the project including the types of tests required, test documents, test methods, and test data collection. Each test from unit through acceptance testing is specified in terms of entrance and exit criteria and the expected level of involvement from the project team, test group, and other functional areas.

Unit and integration tests with appropriate data must be developed to exercise and validate all specified application requirements, functions, and objectives. System and Acceptance tests validate that the integrated system meets the requirements.

Each type of test must use controlled computer generated or live data as specified. The test data must be prepared to include values that will verify the functional capabilities of the software test component, identify its limitations and deficiencies (if any), exercise its capabilities, and verify that the software component performs its intended function as required.

If pilot testing or a phased implementation is required for the software product, the Project Test Plan should include such requirements. In the case of an implementation involving phased software releases, the plan should include the requirements for regression testing of the complete application as new elements are introduced.

For each type of test conducted, the test results are compared with the expected results. Discrepancies are identified and any problems resolved. Retesting is required to verify that the problem solution eliminates the problem and does not introduce new errors. The final test results are accompanied by a completed test results/error log form. This form is completed by the individual(s) responsible for testing and attached to the documents that certify the completion of each type of test.

Task: **4.6.2**
Identify Test Phases

Description: The software product should be tested in four sequential phases: unit, integration, system, and acceptance. Some projects may require additional types of tests (such as prototype testing for offsite installations). The four test phases and prototype testing are described below.

Unit Test Phase: The unit test phase involves testing of the individual software units or groups of related units. A unit is a component that is not subdivided into other components; it is a logically separable part of a computer program. Evaluate each unit of code on how well it meets the performance requirements for which it was designed. Consider timing, memory, accuracy in producing numerical and logical results; and the preparation of input and output required for validating program logic, syntax, and performance requirements. This test phase is performed by the programmer(s) responsible for writing the code.

Integration Test Phase: Integration testing is an orderly progression of testing in which software elements, hardware elements, or both are combined and tested to evaluate the interaction between them. Each program/module must be tested. Integration testing is required to validate that groups of related programs, when combined to establish an integrated functional module of code, interface properly, and perform the software functions for which they were designed. Examine the source program/module statements to ensure that the program logic meets the requirements of the design and that the application satisfies an explicit functional requirement. This test phase is performed by the project team.

System Test Phase: The system test phase tests the integrated hardware and software to verify that the software product meets its specified requirements and operates successfully on the host platform. This test phase is required to validate, when the entire software product is loaded onto the host platform, that the proper initialization is performed; decision branching paths are appropriate; and all software functions are performed as specified in the Software Requirements Specification. System testing validates that the software product produces the required outputs and interfaces properly with other systems with which the software product gives or receives data; that transaction response times meet user expectations; and machine resource allocation and utilization are within expected norms. This test phase can be performed by the project team or by an independent test group with support from the project team.

Acceptance Test Phase: Acceptance testing is conducted to determine whether a software product satisfies its acceptance criteria and to enable the system owner's organization to determine whether to accept the software product. The acceptance test is required to validate that the software, its related documentation, tools, and hardware, satisfy all of the specified requirements and objectives of the system owner's organization, DOE standards, the requirements specification, and the design criteria. Acceptance testing will include tests of all intrasystem interfaces; and the use of all manuals, documentation, procedures, and controls. This test phase can be performed by the project team with system owner and user observers or by system owner and user representatives with support from the project team.

Prototype Testing: In addition to the four test phases, a prototype or site test can be used when software must be physically transported, installed, and made operational at a computer facility other than at the site(s) where the acceptance test was conducted. When required, this test is conducted at selected user location(s) that will totally test the software product under "live" conditions with users and support personnel.

Task: **4.6.3**
Identify Test Environment Requirements

Description: The Project Test Plan should outline what is needed to perform testing activities throughout the project lifecycle including personnel, hardware, software, space, and other environmental requirements. As much testing as possible should be performed on the same equipment that will be used for the production system. In many cases, this information is not fully known until the System Design Stage.

The following are some of the considerations for test environment requirements.

- Evaluate automated testing tools for the following:
  - Generation of test scripts
  - Creation of result and error repositories
  - Consideration of each tool's benefits and costs
  - Use of simulators
- Determine local area network, wide area network, and metropolitan area network testing environment(s), as needed
- Determine test lab, data generation, and error correction support
- Identify Beta test sites

Activity: **4.7**
Develop Acceptance Test Plan

Responsibility: Project Team

Description: The Acceptance Test Plan is a description of the test activities planned for project acceptance. The Acceptance Test Plan should establish the testing necessary to validate that the project requirements have been met and that the deliverables are at an acceptable level in accordance with existing standards. The plan also assures that a systematic approach to acceptance testing is established and that the testing is adequate to verify the functionality of the software product.

The complete set of system requirements and acceptance criteria form the basis for determining the overall approach to acceptance testing and the specific testing and examination methods. Features of the installation site and the software system affect how the software acceptance testing will be done. Unique arrangements may be necessary when the software cannot be completely installed and executed in a live environment. Multiple configurations may have to be distributed at several installation sites.

When a new system is a replacement for one already in use, the acceptance test must assure the integrity of the users business operations while placing the replacement into operation. For example, the old system and the new system are used in parallel until complete functionality has been verified. In some cases, the acceptance process may take several months to assure that a complete business or accounting cycle has occurred. This concern will influence the approach to software acceptance testing.

Work Product: Software acceptance testing must be documented carefully with traceability of test cases to the software requirements and acceptance criteria established by the system owner. As a minimum, the acceptance test plan should address the following requirements.

- Identification of the personnel involved in the acceptance test process and their testing responsibilities. If individuals outside of the project team perform acceptance testing, include the responsibilities and relationships of external test groups.
- Traceability of test designs and cases to software requirements.
- The objectives and constraints for each test.

Work Product, continued:

- Complete test cases and test procedures including inputs and expected outputs for each test case.
- Descriptions of error reporting, analysis, and resolution.
- Location(s) where testing will occur, the testing approach, type of facilities, and tester training.
- Acquisition of special purpose testing equipment, tools, and software.
- Resources and cost estimation to accomplish testing.

Place a copy of the draft Acceptance Test Plan in the Project File. The draft plan will be reviewed during the Software Integration and Testing Stage and delivered as a final document.

Review Process: Conduct structured walkthroughs to assure the draft Acceptance Test Plan adequately describes all testing activities, test schedules, test products, test responsibilities, the testing methodology, and the required resources.

Activity: **4.8**
Select Design Methodology

Responsibility: Project Team

Description: A systematic approach for building the functional and system designs for the software product simplifies the process and results in a software product that is testable, reliable, and maintainable. A complete design methodology includes the following elements:

- A methodology that is compatible with the requirements analysis methodology and any automated tools used by the project team.
- Straightforward rules that relate information obtained during requirements analysis to a distinct software structure.
- Design standards that comply with the site's current software engineering practices, the system owner organization's standards, and the constraints imposed by the software and hardware tools used by the project team.
- A practical approach to design that is amenable to a wide variety of software products.
- The development of small, intermediate design products that can be used to measure quality and progress.
- An evolution process from functional to system design.
- Well-defined measures to assess the quality of the design.
- Guidance on how to detect and correct design features that reduce maintainability and reusability.

The value of a design methodology can be significantly enhanced by automated tools that directly support the methodology. Automated tools provide assistance in generating, maintaining, and analyzing design diagrams and data dictionaries. The use of such tools typically results in a design that is easier to maintain, higher in quality, and more complete than designs produced without automated tools. The increased quality leads to significant productivity gains during software programming and testing.

Sample Design Methods: The following are examples of some common design methodologies.

- Function-oriented design methods model the software product by breaking it into components, identifying the inputs required by those components, and identifying the outputs produced by them. Function-oriented design methods include structured analysis and structured design. The major models or design representations used by this method are data flow diagrams, data dictionaries, structure charts, and process specifications.

- Data-oriented design methods use program structures that are derived from the data structures. Tree diagrams are typically used to represent both the data and the program structures.

- Object-oriented design methods produce a software architecture based on the objects manipulated by systems or subsystems rather than by functions. An object-oriented design closely resembles a model of reality since it captures the real-world objects and the operations taken by or upon them. The design structure tends to be layers of abstraction where each layer represents a collection of objects with limited visibility to other layers.

Work Product: Create a description of the design methodology and distribute it to the project team, system owner, and users. Place a copy of the design methodology description in the Project File.

Review Process: Conduct a structured walkthrough to verify that the design methodology is appropriate for the scope and objectives of the project. A structured walkthrough is not needed when the methodology has been used successfully on similar projects for the same system owner/user computing environment.

Activity: **4.9**
Revise Project Plan

Responsibility: Project Manager

Description: Once the requirements are baselined, determine if the project estimates for resources, cost, and schedule need to be revised and if the selected development approach is still appropriate for the size and complexity of the project.

Work Product: Review the Project Plan for accuracy and completeness of all Requirements Definition Stage activities and make any changes needed to update the information. Expand the information for the Functional 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 Functional Design Stage.

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

Activity: **4.10**
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 Requirements Definition Stage Exit process. Additional ISAs can be performed during the stage, as appropriate. An ISA is recommended after the completion of the Software Requirements Specification.

Provide the reviewer with copies of all work products developed or revised during the Requirements Definition 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: **4.11**
Conduct Requirements Definition 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 Requirements Definition 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 *Stage Exit Process 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 Functional Design Stage.