

# **SANDIA REPORT**

SAND97-2474 • UC-705

Unlimited Release

Printed October 1997

## **Software Development Methodology for High Consequence Systems**

L. S. Baca, J. F. Bouchard, E. W. Collins, M. Eisenhour, D. D. Neidigk,  
M. J. Shortencarier, P. A. Trelue

Prepared by  
Sandia National Laboratories  
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,  
a Lockheed Martin Company, for the United States Department of  
Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



**Sandia National Laboratories**

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from  
Office of Scientific and Technical Information  
PO Box 62  
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from  
National Technical Information Service  
US Department of Commerce  
5285 Port Royal Rd  
Springfield, VA 22161

NTIS price codes  
Printed copy: A08  
Microfiche copy: A01

Distribution Category  
UC-705

SAND97-2474  
Unlimited Release  
Printed October 1997

# Software Development Methodology for High Consequence Systems

L. S. Baca, J. F. Bouchard, E. W. Collins, M. Eisenhour, D. D. Neidigk,  
M. J. Shortencarier, P. A. Trelue

*Sandia National Laboratories  
Albuquerque, New Mexico 87185*

## **Abstract**

This document describes a Software Development Methodology for High Consequence Systems. A High Consequence System is a system whose failure could lead to serious injury, loss of life, destruction of valuable resources, unauthorized use, damaged reputation or loss of credibility or compromise of protected information.

This methodology can be scaled for use in projects of any size and complexity and does not prescribe any specific software engineering technology. Tasks are described that ensure software is developed in a controlled environment. The effort needed to complete the tasks will vary according to the size, complexity, and risks of the project. The emphasis of this methodology is on obtaining the desired attributes for each individual High Consequence System.

## **Foreword**

The Software Development Methodology for High Consequence Systems (SDMHCS) was sponsored by Larry Dalton, Manager of the High Integrity Software Systems Engineering Department at Sandia National Laboratories. This methodology was developed to support the software engineering work being done by that department and the Surety Electronics and Software Department in support of high consequence applications.

Use of this methodology does not automatically ensure software or system surety. Software is only one component within the context of a larger system that includes computer hardware, and possibly other devices (e.g., mechanical, electrical) as well as human operators. Software is only a surety concern within the context of this larger system.

Some of the techniques described within this methodology have been used in software development projects at Sandia National Laboratories.

This methodology was developed by a working group consisting of the following members:

**Lorraine Baca, Chair**

**Julie Bouchard**

**Elmer Collins**

**Marianna Eisenhour**

**David Neidigk**

**Mickey Shortencarier**

**Patty Trelue**

## **Acknowledgments**

This document was not developed in isolation of existing materials documenting similar concepts. The origin of this document was the Sandia Preferred Processes for Software Development, and much of the work of that document is reflected in this methodology. In addition, the working group acknowledges information obtained from various sources in the reference section in Appendix A.

The working group would like to recognize the following individuals for their contributions to the development of this document:

**Laney Chino-Kidd**

**Danielle Clibon**

**Mark Ekman**

**Gerald McDonald**

## DOCUMENT REVISION SHEET

DOCUMENT TITLE: Software Development Methodology for High Consequence Systems

Document Number: TBD

Rev.: Version 3.0

<b>Section No.</b>	<b>Additions:</b>
--------------------	-------------------

<b>Section No.</b>	<b>Deletions:</b>
--------------------	-------------------

<b>Section No.</b>	<b>Changes:</b>
--------------------	-----------------

3.2.7	Added SHA references to Task 3 in the Requirements Phase.
-------	---

3.3.8	Modified task description in Task 7 to include more detailed information. Also added DHA references.
-------	--

3.4.8	Modified task description in Task 2 to include more detailed information. Also added code-level hazard analysis references to Task 2. Modified task description in Task 5 to include more detailed information about surety testing. Also added software surety testing references to Task 5.
-------	---

3.5.8	Added references containing information about software surety testing in Task 3.
-------	--

3.6.8	Modified task description in Task 5 to include more detailed information on surety testing. Also added references containing information about software surety testing.
-------	---

Appendix A	Added references containing information about different types of hazard analysis to the Reference list.
------------	---

## DOCUMENT REVISION SHEET

DOCUMENT TITLE: Software Development Methodology for High Consequence Systems

Document Number: TBD

Rev.: Version 2.0

Section No.	Additions:
before 1.0	Added Executive Summary

Section No.	Deletions:

Section No.	Changes:
Foreword	Modified paragraph referring to Command and Control department changing to High Integrity Software Systems Engineering. Also, deleted statement claiming this methodology has never been applied to a software development project at Sandia. Also, added Surety Electronics and Software Department to first paragraph.
2.	Added page breaks before and after Section 2, Acronyms.
3.1.1	Add sentence to end of first paragraph explaining that “manager” is person responsible for overseeing day-to-day activities.
Global	Changed “Command & Control Department” everywhere to “High Integrity Software Systems Engineering Department”.
3.1.6	In Task Flow, made Task 5 a precedent of Tasks 8 and 9.
3.1.8	In Task 8 and Task 9, changed Dependencies from none to Task 5 to match flow.
3.2.5	In Task Flow, added dotted lines to and from Tasks 5 and 8, also Tasks 3 and 4.
3.2.7	In Tasks 4, 5, and 8, changed dependencies section to agree with task flow. In Task 5, added another input (plan for software system test effort from Task 8). In Task 8, added another input (selected test support tools from Task 5). In Task 8, added test environment to list in the task description section. In Task 3, added test engineer to persons available for review of the SHA (in the Responsibilities section).
3.3.8	In Task 6, removed “graphical” from 1 <sup>st</sup> sentence of the task description allowing for any type of design representation.
3.4.1	In the overview section of the Implementation Phase, removed the words “cleanly compiled”, from the first sentence. Unit testing automatically implies cleanly compiled.
3.5.1	In the overview section of the Integration Phase, added sentence to second paragraph stating

	that if hardware is available, integration testing should include testing hardware/software interfaces.
3.5.3	In Integration Phase Output Table 3-15, added “Source Code” as an output of this phase.
3.5.6	In Task Flow, added dotted lines between Task 3 and Task 4.
3.5.8	In Task 3 and Task 4, changed Dependencies section to match flow. Also added SDD and SRS to the Inputs section. In Task 5, added SRS to Support Materials section. In Task 3, added Step 4 to Task Description to account for code corrections needed when errors were found. Also, in Task 3, added source code as an output from this task.
3.6.3	In Test Phase Output Table 3-18, added “Source Code” as an output of this phase.
3.6.8	In the Test Phase, Task 2, added SRS to the Support Materials section. In Tasks 5 and 7, added activity to the Task Descriptions which describes that the software developer should make all necessary source code corrections due to errors found. Also, in both Tasks 5 and 7, added “updated Source Code” as an output of this task.
Appendix B	Added terms “Software Development Folder” and “Surety Folder” to list of definitions.
Global	Replaced “unit code” with “source code” everywhere.

# Table of Contents

<b>1. Introduction.....</b>	<b>3</b>
<b>1.1 Problem Statement.....</b>	<b>3</b>
<b>1.2 Purpose .....</b>	<b>3</b>
<b>1.3 Scope .....</b>	<b>3</b>
<b>1.4 Audience .....</b>	<b>3</b>
<b>1.5 Document Overview .....</b>	<b>4</b>
1.5.1 Section Summary .....	4
1.5.2 Maintenance.....	6
<b>2. Acronyms.....</b>	<b>7</b>
<b>3. Software Development Phases for High Consequence Systems .....</b>	<b>9</b>
<b>3.1 Project Planning Phase.....</b>	<b>10</b>
3.1.1 Overview.....	10
3.1.2 Inputs and Suppliers.....	10
3.1.3 Outputs and Customers .....	13
3.1.4 Metrics .....	13
3.1.5 Task Summary .....	13
3.1.6 Task Flow .....	15
3.1.7 Modification Requests .....	16
3.1.8 Detailed Task Descriptions .....	17
<b>3.2 Requirements Phase.....</b>	<b>38</b>
3.2.1 Overview.....	38
3.2.2 Inputs and Suppliers.....	38
3.2.3 Outputs and Customers .....	39
3.2.4 Metrics .....	39
3.2.5 Task Flow .....	41
3.2.6 Modification Requests .....	42
3.2.7 Detailed Task Descriptions .....	43
<b>3.3 Design Phase.....</b>	<b>54</b>
3.3.1 Overview.....	54
3.3.2 Inputs and Suppliers.....	54
3.3.3 Outputs and Customers .....	54
3.3.4 Metrics .....	55
3.3.5 Task Summary .....	55
3.3.6 Task Flow .....	56
3.3.7 Modification Requests .....	57
3.3.8 Detailed Task Descriptions .....	58
<b>3.4 Implementation Phase .....</b>	<b>73</b>
3.4.1 Overview.....	73
3.4.2 Inputs and Suppliers.....	73
3.4.3 Outputs and Customers .....	73
3.4.4 Metrics .....	74
3.4.5 Task Summary .....	74

3.4.6 Task Flow .....	75
3.4.7 Modification Requests .....	76
3.4.8 Detailed Task Descriptions .....	77
<b>3.5 Integration Phase .....</b>	<b>83</b>
3.5.1 Overview.....	83
3.5.2 Inputs and Suppliers.....	83
3.5.3 Outputs and Customers .....	84
3.5.4 Metrics .....	84
3.5.5 Task Summary .....	84
3.5.6 Task Flow .....	85
3.5.7 Modification Requests .....	86
3.5.8 Detailed Task Descriptions .....	87
<b>3.6 Test Phase .....</b>	<b>93</b>
3.6.1 Overview.....	93
3.6.2 Inputs and Suppliers.....	93
3.6.3 Outputs and Customers .....	94
3.6.4 Metrics .....	94
3.6.5 Task Summary .....	94
3.6.6 Task Flow .....	95
3.6.7 Modification Requests .....	96
3.6.8 Detailed Task Descriptions .....	97
<b>4. Closing Remarks .....</b>	<b>106</b>
4.1.1 Overview.....	106
4.1.2 Completion of System Development .....	106
4.1.3 Project Archival and Maintenance .....	106
4.1.4 Verification and Regulatory Agencies .....	107
4.1.5 Metrics .....	107
<b>5. Support Processes .....</b>	<b>108</b>
<b>5.1 Inspection Process.....</b>	<b>108</b>
5.1.1 Overview.....	108
5.1.2 Inputs and Suppliers.....	108
5.1.3 Outputs and Customers .....	109
5.1.4 Metrics .....	109
5.1.5 Task Summary .....	109
5.1.6 Task Flow .....	110
5.1.7 Detailed Task Descriptions .....	111
<b>5.2 Configuration Management Process .....</b>	<b>116</b>
<b>5.3 Software Quality Assurance Process.....</b>	<b>118</b>
<b>6. Conclusions and Recommendations .....</b>	<b>120</b>
<b>6.1 Future Enhancements to this Document .....</b>	<b>120</b>
<b>Appendix A - References .....</b>	<b>121</b>
<b>Appendix B - Definitions .....</b>	<b>123</b>
<b>Appendix C - Overview of Formal Methods .....</b>	<b>126</b>
<b>Appendix D - Surety Folder Contents.....</b>	<b>127</b>

## Executive Summary

The Software Development Methodology for High Consequence Systems (SDMHCS) sponsored by the High Integrity Software Systems Engineering Department at Sandia National Laboratories, was developed to support the software engineering work for high consequence applications being performed by that department and the Surety Electronics and Software Department. These departments have developed methods/techniques, documented processes, and personnel expertise in the general area of high integrity software engineering. These processes and capabilities are continually being improved in order to achieve their vision of providing the necessary capabilities to develop cost effective, high surety solutions for High Consequence Systems (HCSs). Derived from these processes and capabilities, specific approaches to the development of surety-critical software are documented in this methodology.

This document provides a set of processes with a detailed list of tasks for each phase in the software development lifecycle. It is specifically aimed at the development and assurance of HCS software and intended for use by software developers at Sandia. The phases in a software development lifecycle include software project planning, software requirements, software design, implementation, software integration, and software test. The origin of this document was the Sandia Preferred Processes for Software Development.

The Project Planning Phase is the initial phase in the Software Development Methodology for HCSs. It contains the tasks necessary for both software development planning and test planning. It is the intent of this phase to parallel software development planning with test planning. By coordinating these two seemingly independent efforts early on, this methodology stresses the prevention of defects while maintaining the detection of defects as a secondary goal. This phase outputs a Software Development Plan (SDP) and a Master Test Plan (MTP).

The Software Requirements Phase is typically considered the first formal mandatory phase of software development. This phase takes a high-level product description and often ambiguous and untestable customer requirements and outputs a Software Requirements Specification (SRS). In particular, emphasis is placed on the surety requirements of the system and a Software Hazard Analysis is performed. Other phases use the SRS as the fundamental requirement guide. The Requirements Phase does not include the definition or specification of internal system constraints, limitations, or design guidelines. The requirements specification should describe *what* will be developed, not *how* it will be developed. In addition to the SRS, testing activities are performed in preparation for testing activities executed in later phases.

The purpose of the Design Phase is to produce a blueprint for the implementation of the software product and is documented in a Software Design Description (SDD). The design described in the SDD must satisfy all requirements stated in the SRS. The Design Phase produces the definition or specification of internal system constraints, limitations and design guidelines. In this phase, a Design Hazard Analysis is performed to ensure that surety requirements have been properly defined in the design and that no new hazards have been introduced. In addition, this phase produces the software system test cases and an integration test plan.

The Implementation Phase uses the blueprint provided by the Design Phase to produce code units which have been thoroughly unit tested. Inspection of the code includes a surety review which assures that the code correctly implements the intent of the surety requirements and that no additional hazards have been introduced. The Implementation Phase provides the code units to the Integration Phase where the units are combined and tested.

The purpose of the Integration Phase is to determine how individual software units perform together as an integrated system. Depending on the size of the system, there may be several levels of integration tests. For example, units are first integrated into components, components are integrated into subsystems, and then subsystems are combined to form the total software system. It is not the purpose of the integration tests to revalidate requirements previously verified during lower-level tests but rather to build on previous test and evaluation. Integration testing should focus on the internal interfaces between software units. In this phase, a surety testing matrix is created which maps surety requirements to a testing approach. The Integration Phase provides input primarily in the form of a Software System Test Plan (SSTP) to the Test Phase.

The purpose of the Test Phase is to provide complete test coverage of all the software requirements stated in the SRS. Special emphasis must be placed on surety requirements to ensure that surety related tests have been identified for each surety requirement and all surety-related test cases have been identified and documented. The Test Phase

may also test hardware/software interactions and software/software interactions. The Test Phase provides the updated SSTP to the customer and verification personnel as evidence that functional and surety requirements have been tested. The SSTP also identifies a set of regression tests that can be run on the software to ensure that modifications to the software have not inadvertently caused a defect.

## **1. Introduction**

This document contains a software development methodology that can be applied to High Consequence Systems (HCSs). An HCS is a system whose failure could lead to serious injury, loss of life, destruction of valuable resources, unauthorized use, damaged reputation or loss of credibility, or compromise of protected information. This methodology prescribes the tasks which must be completed in each phase of software development as well as the order in which the tasks must be completed. It does not prescribe any specific software engineering technologies for use in this methodology, rather any technique should be compatible with this methodology.

### **1.1 Problem Statement**

As more and more critical functions in HCSs are being controlled by software, it is becoming increasingly important that this software be developed in a controlled environment. The desired attributes of an HCS are that it:

- Works properly
- Is well-defined with no random behavior
- Fails safely
- Is only accessible to authorized personnel
- Is available, reliable, and maintainable
- Is quantitatively measurable
- Contains trusted components
- Minimizes dependencies on the user(s)
- Has acceptable risks associated with its use

### **1.2 Purpose**

This document describes the phases of a software development methodology for HCSs. This methodology can be scaled for use in projects of any size. The effort needed to complete the tasks within the phases will vary according to the size, complexity, and risks of the project. Certain activities and tasks may not be applicable for some projects. After evaluating the attributes of a project, developers should feel free to add or delete tasks of this methodology, as appropriate.

### **1.3 Scope**

The scope of this document consists of activities and tasks relating to software development life cycle phases. The phases included in this document are: project planning, requirements, design, implementation, integration, and test. Although mentioned in a general sense throughout this document and in Sections 5.2 and 5.3, this document does not address software configuration management activities, software quality assurance activities, or software maintenance activities in detail.

### **1.4 Audience**

The proposed audience for this document is employees of Sandia who are involved in developing software for HCSs. This methodology is designed to be applicable for development of research or applications software, software in any language using any design technique, and large or small software systems.

Although developed with an emphasis on activities relating to HCSs, this methodology can be tailored for use by any software development effort.

## 1.5 Document Overview

### 1.5.1 Section Summary

Section 1 of this document is the Introduction section and contains the Problem Statement, Purpose, Scope, Audience, and Document Overview subsections.

Section 2 of this document is the Acronyms section and contains definitions of those acronyms that are used in this document.

Section 3 of this document is the Software Development Phases for HCSs. It contains subsections for the following phases of software development:

- Project Planning
- Requirements
- Design
- Implementation
- Integration
- Test

Section 4 contains closing remarks for the SDMHCS. This section briefly discusses project completion activities such as completion of system development, project archival and maintenance, verification activities, and metrics analysis.

Section 5, Support Processes, contains process descriptions for the Inspection Process, Configuration Management Process, and Software Quality Assurance Process. These three processes are integrated with the software development phases identified in this document.

The subsections in Section 3 and the Inspection Process of Section 5 contain descriptions of tasks associated with that phase or process description. An outline of the contents of each subsection is provided in Table 1-1 below.

These tasks interface with each other via required inputs and outputs. If Task A produces an output which is required by Task B, then Task B is recognized as an internal customer of Task A. Therefore, developers working in Task A should treat developers working in Task B as their customers. Internal customer satisfaction leads to a higher quality product.

For each phase or process description found in Section 3 and the Inspection Process of Section 5, a given task cannot be completed until all tasks upon which it is dependent are complete. The given task should not even begin until all of the “dependent” tasks are complete. However, while working on a given task, defects or omissions may be discovered which require that dependent tasks be reopened and reworked. Work on the given task should cease until rework of the dependent tasks are complete.

**Table 1-1: Outline of Software Phases and Support Processes Sections**

<b>Overview</b>	The overview describes the function of this phase within a software development life cycle. It indicates the impact of the phase on the entire development effort and describes how the phase meets the requirements for succeeding phases.
<b>Inputs and Suppliers</b>	This section describes the inputs necessary for a software engineer to begin the phase or complete its assigned tasks. It includes a table showing the required inputs to the phase and the suppliers of these inputs.
<b>Outputs and Customers</b>	This section describes the outputs necessary before the phase can be completed. A table is included that shows the outputs and the customers for these outputs.
<b>Metrics</b>	This section describes metrics used to measure how well the phase and the phase suppliers are meeting requirements.
<b>Task Summary</b>	A sequential list of the required tasks for the phase and their descriptions are given in this section. It also identifies the prior tasks required to perform a given task.
<b>Task Flow</b>	This section provides a graphical representation of the order required for execution of all tasks defined by this phase/process.
<b>Modification Requests</b>	In moving through the software development life cycle, events occur that require changes to be made to tasks already completed. This section defines how modification requests might be generated by the phase and how the phase handles the receipt of modification requests.
<b>Detailed Task Descriptions</b>	<p>Each task of the phase is described in this section. The detailed task descriptions are presented in the following format.</p> <p><b>Objectives-</b> The purpose of the task.</p> <p><b>Dependencies-</b> Identification of the tasks which must be completed before this task is begun.</p> <p><b>Responsibilities-</b> A statement of required responsibilities for the task.</p> <p><b>Inputs-</b> Information and products created prior to the task which are used by the task.</p> <p><b>Entrance Criteria-</b> Conditions that must be satisfied before the task can begin.</p> <p><b>Task Description-</b> Steps which must be performed to complete the task.</p> <p><b>Exit Criteria-</b> Conditions that must be satisfied before the task can end.</p> <p><b>Outputs-</b> The minimum set of deliverables resulting from the task.</p> <p><b>References-</b> The standards, guidelines, and other documentation that may be useful in meeting the objective of this task.</p>

Section 6 completes the document with Conclusions and Recommendations.

Appendix A contains all references used in this document.

Appendix B contains definitions for terms used throughout this document including terms related to HCSs.

Appendix C contains an overview of formal methods.

Appendix D contains a description of the contents of the Surety Folder.

### **1.5.2 Maintenance**

This methodology is a living document and will be improved iteratively through experience and metrics analysis. The responsibility for the maintenance of this document will reside with the High Integrity Software Systems Engineering Department and will be implemented through their Department Quality Plan.

## 2. Acronyms

CASE	Computer-Aided Software Engineering
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
CM	Configuration Management
COTS	Commercial-Off-The-Shelf
DHA	Design Hazard Analysis
DOE	Department of Energy
HCS	High Consequence System
IDD	Interface Design Document
IEEE	Institute of Electrical and Electronics Engineers
IRS	Interface Requirements Specification
MIL-STD	Military Standard
MTP	Master Test Plan
PHA	Preliminary Hazard Analysis
RTM	Requirements Trace Matrix
RMP	Risk Management Plan
SCMP	Software Configuration Management Plan
SDD	Software Design Description
SDF	Software Development Folder
SDMHCS	Software Development Methodology for High Consequence Systems
SDP	Software Development Plan
SEE	Software Engineering Environment
SEI	Software Engineering Institute
SHA	Software Hazard Analysis
SLP	Sandia Laboratories Policy
SMP	Software Management Plan
SOW	Statement Of Work
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
SRS	Software Requirements Specification
SSP	Software Safety Plan
SSTP	Software System Test Plan
STP	Software Test Plan

SW	Software
TBD	To Be Determined
VDM	Vienna Development Method
WBS	Work Breakdown Structure

### 3. Software Development Phases for High Consequence Systems

This section specifies the relationship among specific software development tasks within the framework of standard software life cycle phases. Another way to view the software development tasks is to look at them within the framework of development activities, e.g., planning, requirements, testing. Table 3-1 shows the relationship between the development phases, life cycle activities, and the software development tasks identified in this document.

**Table 3-1: Relationship of Life Cycle Phases to Development Activities**

<b>Life Cycle Dev. Phase Activities</b>	<b>Project Planning Phase</b>	<b>Requirements Phase</b>	<b>Design Phase</b>	<b>Implementation Phase</b>	<b>Integration Phase</b>	<b>Test Phase</b>
<b>Project Planning Activities</b>	Tasks 1-14	Task 5				
<b>Requirements Activities</b>	Task 10	Tasks 1-9				
<b>Design Activities</b>	Task 10		Tasks 1, 4-6, 8, 9			
<b>Implementation Activities</b>				Tasks 1-5		
<b>Integration Activities</b>			Tasks 10, 11		Tasks 1-5	
<b>Test Activities</b>	Tasks 6, 8, 10-14	Tasks 5, 8	Tasks 1, 2, 3		Tasks 4, 5	Tasks 1-8
<b>Verification Activities</b>	Task 13	Task 6	Tasks 3, 9, 11	Task 4		Task 2
<b>Surety Activities</b>	Tasks 2-8	Tasks 2-5	Task 2, 3, 4, 7, 9, 11	Tasks 1, 2	Task 4	Tasks 1, 5, 6, 7

## 3.1 Project Planning Phase

### 3.1.1 Overview

The Project Planning Phase is the initial phase in the Software Development Methodology for HCSs (SDMHCS). It contains the tasks necessary for both software development planning and test planning. The Project Planning Phase takes as input a system Preliminary Hazard Analysis (PHA), system-level requirements, and surety requirements. It outputs a Software Development Plan (SDP) and a Master Test Plan (MTP). The MTP may be documented as part of the SDP. This phase also outputs several other planning documents (e.g., schedule, spend plan, risk management plan, software configuration management plan, and software quality assurance plan) all of which could be documented as part of the SDP. It is assumed that a system project manager, a software project manager, a software test manager, system surety experts, and system engineers are available prior to the start of this phase. The term “manager” refers to the person who oversees the day-to-day tasks of the project (not necessarily the budget manager).

It is the intent of this phase to parallel software development planning with test planning. By coordinating these two seemingly independent efforts early on, this methodology stresses the prevention of defects while maintaining the detection of defects as a secondary goal.

The Project Planning Phase provides input primarily in the form of an SDP and an MTP to the following phases:

- Requirements
- Design
- Implementation
- Integration
- Test

The SDP serves as the fundamental development guide for the entire software life cycle. The SDP defines the technical and managerial processes necessary to satisfy the project requirements as defined by the customer and specifies the software development activities necessary to ensure that the software product meets all specified design requirements and results in a quality product. It is the contract between the software project manager and the software developers as well as the project team members and the customer. In addition, for high consequence software, the SDP must define all surety-related processes and activities necessary to produce a highly-reliable, safe, and/or secure system.

The MTP serves as the fundamental testing document that controls and guides all testing efforts throughout the entire software life cycle. The MTP defines the relationship between unit, integration, and system level testing and maintains a consistent approach across components of a system. The MTP defines the overall testing strategy needed to satisfy the project testing requirements as defined by the customer and/or any verification agencies and specifies the testing activities necessary to ensure that the software product meets all specified design requirements and results in a quality product. It is the contract between the software test manager and the development team as well as the test team members and the customer. In addition, for high-consequence software, the MTP must define all supplementary surety-related testing processes and activities necessary to produce a highly-reliable, safe, and/or secure system.

### 3.1.2 Inputs and Suppliers

To begin the Project Planning Phase for an HCS, a perceived or established need for a given product must exist and the software project manager must begin defining software development approaches, strategies, and processes regarding surety as a critical factor. In addition, the software test manager must begin defining test approaches, strategies, and processes also regarding surety as a critical factor.

The Project Planning Phase for an HCS must accept a wide range of inputs from a wide range of suppliers unlike many of the other development phases that have a narrow input interface with their previous input sources. Because

## Project Planning Phase

the Project Planning Phase is the initial phase in this methodology, this phase interfaces with suppliers for which specific input criteria cannot be enforced. For a system that is judged to be an HCS, a Preliminary Hazard Analysis (PHA) should be performed on the high-level system conceptualization by a surety team comprised of the customer/user, system designers, and surety experts from the areas of concern. The PHA should identify all surety-critical functions of the system. A surety-critical function is a function that allows the system to enter a hazardous state, that moves the system from a hazardous state to a non-hazardous state, or that mitigates the consequences of the system entering a hazardous state. The PHA should also define all the hazards, the criticality of each hazard, and the resources (administrative controls, hardware, or software) dedicated to control or mitigate the hazard.

Table 3-2 lists the possible inputs to the Project Planning Phase and their suppliers

Project Planning Phase

**Table 3-2 - Project Planning Phase Inputs and Suppliers**

<b>Input</b>	<b>Possible Supplier(s)</b>
Customer's Existing Environment	Customer
Hardware/Software Interdependencies	Customer
Higher Level Test and Evaluation Plans	Customer, Project Manager
Lessons Learned from Past Projects	Project Manager, Department Library
Modification Requests	Various Sources
Preliminary Hazard Analysis (PHA)	Customer System Engineers System Surety Experts
Priority Assessment	Customer Project Manager
Regulatory Requirements	Regulatory Agencies, Independent Verification Organizations
Risk Management Plan (RMP)	Project Manager, Corporate Policy, Organization Processes, Department Processes, Technical Leaders
SDP Templates, MTP Templates, SRS Templates, SDD Templates, SSTP Template, SCMP Templates, SQAP Templates	Industry Standards, Corporate Policy, Department Processes
Software Configuration Management Plan (SCMP)	Project Manager, Corporate Policy, Organization Processes, Department Processes
Software Management Plan (SMP)	Project Manager, Corporate Policy, Organization Processes, Department Processes
Software Quality Assurance Plan (SQAP)	Project Manager, Corporate Policy, Organization Processes, Department Processes
Software Standards and Processes	Organization Processes, Department Processes
Statement of Work (SOW)	Customer
System Specifications (Allocated system/software requirements)	Customer, System Engineers
System Surety Requirements	Customer, System Engineers, System Surety Experts
System Work Breakdown Structure (WBS)	Customer, Project Manager
System-Level Project Plan and Schedule	Customer, Project Manager
Technology Needs	Project Manager
User's Manual	Customer

### 3.1.3 Outputs and Customers

Table 3-3 contains the outputs from this phase.

**Table 3-3 - Project Planning Phase Outputs and Customers**

<b>Output</b>	<b>Customer</b>
Modification Requests	Various Phases
Requirements Tracing Method Requirements Tracing Tool	Requirements Phase Design Phase Implementation Phase Integration Phase Test Phase
SDD Template	Design Team Design Phase
SRS Template	Requirements Analysis Team Requirements Phase
SSTP Template	Test Team Requirements, Design, Implementation, Integration, and Test Phases
SDP SCMP SQAP RMP MTP	Customer Software Development Team Software Testing Team Requirements Phase Design Phase Implementation Phase Integration Phase Test Phase Quality Assurance

### 3.1.4 Metrics

The Project Planning Phase begins tracking the following metrics:

- estimated source lines of code,
- number of changes in customer requirements,
- planned and actuals for cost,
- planned and actuals for resources, and
- planned and actuals for schedule.

### 3.1.5 Task Summary

The Software Development Planning Phase tasks are summarized in Table 3-4. Each of these tasks is described in greater detail in Section 3.1.8.

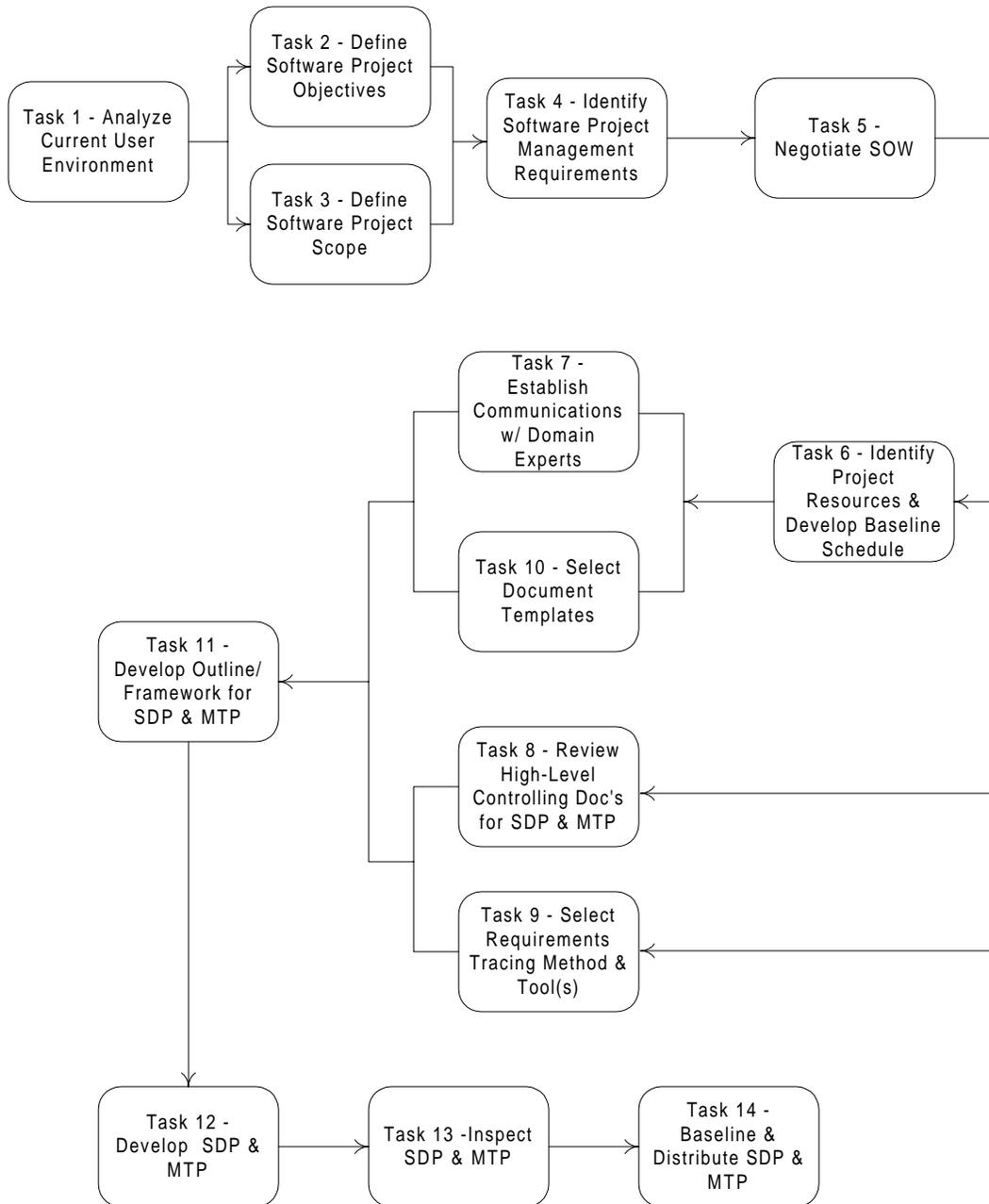
Project Planning Phase

**Table 3-4 - Project Planning Phase Task Summary**

<b>Task</b>	<b>Description</b>	<b>Dependencies</b>
1	Analyze Current User Environment	
2	Define Software Project Objectives	1
3	Define Software Project Scope	1
4	Identify Software Project Management Requirements	2, 3
5	Negotiate Statement Of Work (SOW)	4
6	Identify Project Resources and Develop Baseline Schedule for Software Development and Testing	5
7	Establish Communications With Domain Experts	6
8	Review High-Level Controlling Documents for the SDP and MTP	
9	Select Requirements Tracing Method and Tool(s)	
10	Select Document Templates	6
11	Develop Outline/Framework for SDP and MTP	7, 8, 9, 10
12	Develop SDP and MTP	11
13	Inspect SDP and MTP	12
14	Baseline and Distribute SDP and MTP	13

## Project Planning Phase

### 3.1.6 Task Flow



**Figure 3-1 Project Planning Phase Task Flow**

### **3.1.7 Modification Requests**

#### **3.1.7.1 Generation of Requests**

There are several tasks performed during the Project Planning Phase in which modification requests may be generated which affect the outputs of other phases. For instance, a change in the schedule or resource allocations mandated by an external source would necessitate the project to be re-scoped. In this instance, functionality may be affected which in turn affects all subsequent phase documentation which has been completed. The SRS, SDD, source code, and so forth would require modifications.

#### **3.1.7.2 Receipt and Processing of Requests**

The Project Planning Phase may receive modification requests in order to update an output from this phase. These modification requests are handled by revisiting appropriate tasks from the Project Planning Phase. When tasks are revisited, the time devoted to each one will depend on the scope of the modification request. Tasks which may have taken weeks to perform during the initial completion of the Project Planning Phase may take only hours to perform when addressing a modification request.

Depending on the modification request to the SDP or MTP, only certain tasks or parts of tasks are performed. A brainstorming meeting is held between the software project manager and the software test manager in order to explore the impact that the modification request makes on the SDP and/or the MTP. Possible solutions are explored, a final decision is made by the software project manager, and the SDP and/or MTP are updated. As the project progresses, the most likely modification request will be in the form of a change to the schedule, the allocated resources, or both.

Task 13 is performed with the scope of the review reduced to the changes made to the SDP and the MTP. If the changes are minor, then an informal review may be performed instead of a formal inspection at the discretion of the software project manager and/or software test manager.

### 3.1.8 Detailed Task Descriptions

This section provides the detailed description of each task listed in Table 3-4. Each task should be completed to successfully carry out the Project Planning Phase of development.

#### **Task 1: Analyze Current User Environment**

<b>Objectives</b>	The purpose of this task is to obtain a complete understanding of the current user environment. This understanding is needed to define the objectives, scope, constraints, and high-level requirements of the project.
<b>Dependencies</b>	None.
<b>Responsibilities</b>	The project manager and the software project manager are responsible for completion of this task.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• PHA (at the system level)</li><li>• SOW</li></ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"><li>• Customer's existing environment</li><li>• Hardware/Software interdependencies</li><li>• User's Manual</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	The project manager and the software project manager analyze the user's current environment to understand what the users do, how they do it, and what improvements are desired or needed. This includes gaining an understanding of the functions performed, identifying interfaces within the processes, identifying hazards, and listing process inputs and outputs. Appropriate data collection techniques such as surveys, interviews, and reviews of available system documentation are used to gather data and analyze the user environment. The types of information collected includes mission, work processes, workload, incident reports, accident reports, customer perceptions of hazards, processing/data flow, integration/interfaces, costs, equipment, existing software, and user information. The results are documented and used by subsequent tasks in this phase.
<b>Exit Criteria</b>	Analysis of the current user environment is completed and documented. Any inconsistencies between the PHA and the perceived hazards in the user environment are resolved.
<b>Outputs</b>	The output from this task is the documentation of the current user environment.
<b>References</b>	None.

## **Task 2: Define Software Project Objectives**

<b>Objectives</b>	The purpose of this task is to identify, in measurable terms, what the project is intended to accomplish and why it is being undertaken. All testing objectives are identified in parallel with the identification of the project's software objectives. In addition, the identification of all surety objectives is emphasized.
<b>Dependencies</b>	This task is dependent on Task 1 and interdependent on Task 3.
<b>Responsibilities</b>	The software project manager and the software test manager are responsible for completion of this task. The system surety expert is consulted for an independent review.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• Documentation of customer's existing environment from Task 1</li><li>• SOW</li></ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"><li>• Priority Assessment</li><li>• Regulatory Requirements</li><li>• System Specifications</li><li>• System Surety Requirements</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>The software project manager and the software test manager use the information gathered in Task 1 to define and document both the project's software and testing objectives. Some items that might be considered are: general intent of the product, surety goals that the product supports, organizational functions that the product supports, major functional components of the product, mission(s) that the product supports, strategic goals that the product supports, and anticipated benefits of the product. A priority assessment, if available, is used as the basis for prioritizing the project objectives.</p> <p>The software test manager defines and documents the testing levels to be covered by the MTP and the objectives for each level. In addition, the software test manager defines and documents the major testing activities (test analysis, test design, test implementation, test execution, and test assessment) to be performed at each level as well as the project's surety testing objectives.</p>
<b>Exit Criteria</b>	Analysis of the project's software and testing objectives is completed, prioritized, and documented.
<b>Outputs</b>	<p>The outputs from this task are:</p> <ul style="list-style-type: none"><li>• project's software development objectives,</li><li>• project's surety objectives,</li><li>• project's testing objectives, and</li><li>• project's surety testing objectives.</li></ul> <p>These outputs will be incorporated into the SDP and the MTP.</p>
<b>References</b>	<ul style="list-style-type: none"><li>• Master Test Plan Questionnaire [22].</li></ul>

**Task 3: Define Software Project Scope**

<b>Objectives</b>	The purpose of this task is to analyze and document the project's scope and constraints relating to software development and testing. The project scope details the customer processes, organizations, and functions that are affected by the software product. A thorough understanding of the scope of the project is necessary to determine the project's feasibility given the constraints of cost, schedule, and available resources.
<b>Dependencies</b>	This task is dependent on Task 1 and interdependent on Task 2.
<b>Responsibilities</b>	The software project manager and the software test manager are responsible for completion of this task.
<b>Inputs</b>	<p>The input to this task is:</p> <ul style="list-style-type: none"><li>• SOW</li></ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"><li>• Higher level test and evaluation plans</li><li>• SMP</li><li>• System Specifications</li><li>• System Surety Requirements</li><li>• System-level Project Plan and Schedule</li><li>• Technology needs</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>The software project manager and the software test manager possibly use an interview/survey type of approach with the customer to determine the project scope. Some items that might be considered are: number of people using the product, the facilities where the product is intended to be used, the product's output, estimated life expectancy of the product, surety concerns of the product, the mission-criticality of the product, and disaster recovery requirements of the product. The software project manager documents the project's scope with respect to software development for later incorporation into the SDP.</p> <p>The software test manager uses the information gathered in Task 2 (testing levels and activities) as well as information from higher-level test and evaluation plans, if available, to detail the scope of activities for each level of testing. This information is documented and will be incorporated into the MTP.</p> <p>At this point, the software project's feasibility is examined. If, given the current scope, proposed schedule, available resources, and technology needs it is determined that the project is not feasible, the project may have to be modified or terminated.</p>
<b>Exit Criteria</b>	Project scope is completed and documented. Project is deemed feasible at this point.
<b>Outputs</b>	The output from this task is a formal statement of the project's scope (with respect to software development and testing) with an emphasis on identification and documentation of the surety concerns of the product. This information will later be incorporated into the SDP and the MTP.
<b>References</b>	None.

**Task 4: Identify Software Project Management Requirements**

<b>Objectives</b>	<p>The purpose of this task is to identify and estimate the project management requirements with respect to software development and testing.</p> <p>The project management requirements are sufficiently detailed to re-evaluate the project's feasibility, to estimate the resources needed, and to assess higher-level system software requirements.</p>
<b>Dependencies</b>	<p>This task is dependent on Tasks 2 and 3.</p>
<b>Responsibilities</b>	<p>The software project manager and the software test manager are responsible for completion of this task.</p>
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• Project objectives as documented in Task 2</li><li>• Project scope as documented in Task 3</li><li>• SOW</li></ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"><li>• System Specifications</li><li>• System Surety Requirements</li></ul>
<b>Entrance Criteria</b>	<p>The inputs must be available before this task can start.</p>
<b>Task Description</b>	<p>The project management requirements for software development and testing are organized into categories. The following list of items provides possible categories for consideration:</p> <p>Software Development:</p> <ul style="list-style-type: none"><li>• project organization</li><li>• staffing and training needs</li><li>• deliverables</li><li>• inputs/outputs</li><li>• surety considerations</li><li>• independent assessment activities of surety attributes</li><li>• communications</li><li>• interfaces</li><li>• project reviews</li><li>• software development library</li><li>• development environment including any Commercial-Off-The-Shelf (COTS) products that might be needed</li><li>• problem/change reporting</li><li>• test phase incident resolution</li><li>• test phase surety impact incident resolution</li><li>• software engineering activities</li><li>• qualification testing</li><li>• software product evaluation</li></ul>

## Project Planning Phase

- software development metrics
- project's major activities and milestones
- risk management
- configuration management
- quality assurance

### Testing:

- test configurations
- planning risks and contingencies
- features to be tested
- features not to be tested
- testing approach - levels and strategies
- termination criteria and resumption requirements
- test deliverables
- testing tasks
- test environment and test tools
- staffing and training needs
- summary of responsibilities
- project's major test activities and milestones, i.e., test schedule

After identification of the project management requirements for software development and testing has been completed, higher-level system software requirements are assessed. The software project manager identifies special skills required by the software development activities, the specific application, or the software development environment and specifies staffing needs by role and skill level.

The software test manager identifies special skills required by the testing activities, the specific application, or the test environment and specifies staffing needs by role and skill level.

At this point, the project's feasibility is re-examined. If, given the current high-level project requirements, it is determined that the project is not feasible, the project may have to be modified or terminated.

<b>Exit Criteria</b>	A formal statement of the software project management requirements is completed and documented. Project is deemed feasible at this point.
<b>Outputs</b>	<p>The output from this task is the formal documentation of the project management requirements.</p> <p>This information will later be incorporated into the SDP and the MTP.</p>
<b>References</b>	Sandia Software Guidelines, Vol. 2 [13]

## Project Planning Phase

### **Task 5: Negotiate Statement of Work (SOW)**

<b>Objectives</b>	The purpose of this task is to review and approve the SOW. A thorough understanding of the customer's requirements as documented in the SOW is necessary in order to continue planning of the project. All SOW items must be fully understood and negotiated with the customer. In addition, all surety-related requirements in the SOW must be identified and/or isolated to ensure proper handling of these items.
<b>Dependencies</b>	This task is dependent on Task 4.
<b>Responsibilities</b>	The software project manager, software test manager, systems engineer, line management, system surety expert, and the customer are responsible for completion of this task. The customer is responsible for providing and revising the SOW.
<b>Inputs</b>	The inputs to this task are: <ul style="list-style-type: none"><li>• PHA</li><li>• Project management requirements (from Task 4)</li><li>• SOW</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	The customer submits the SOW prior to this task for independent inspection by each of the SOW reviewers. A meeting with the customer is scheduled, and the SOW inspection team reviews the SOW line by line with the customer. The high-level requirements documented in Task 4 are discussed and should coordinate with the SOW requirements. All surety-related items are identified and discussed in detail. Any modifications to the SOW are agreed upon by all parties and the customer re-submits the SOW for approval after all modifications are complete.
<b>Exit Criteria</b>	Line management and the software project manager approve the revised SOW.
<b>Outputs</b>	The output from this task is the SOW.
<b>References</b>	None.

**Task 6: Identify Project Resources and Develop Baseline Schedule for Software Development and Testing**

<b>Objectives</b>	<p>The purpose of this task is to:</p> <ul style="list-style-type: none"><li>• identify and allocate all project resources needed for development of an HCS software system,</li><li>• develop a detailed baseline schedule for the project, and</li><li>• estimate training needs.</li></ul> <p>In addition to software development and testing personnel, all domain experts are identified and allocated. The identification of surety-related personnel to perform the independent surety assessment is a crucial component of this task.</p>
<b>Dependencies</b>	<p>This task is dependent on Task 5.</p>
<b>Responsibilities</b>	<p>The software project manager, software test manager, project manager, and line management are responsible for completion of this task.</p>
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• PHA</li><li>• Project management requirements (from Task 4)</li><li>• SOW (as modified per Task 5)</li></ul> <p>In addition, the following input may be included if appropriate:</p> <ul style="list-style-type: none"><li>• Software Standards and Processes for Project Planning and Estimating</li><li>• SCMP</li><li>• SQAP</li><li>• System-level project plan and schedule</li><li>• Technology Needs</li><li>• WBS</li></ul>
<b>Entrance Criteria</b>	<p>The inputs must be available before this task can start.</p>
<b>Task Description</b>	<p>Using a defined process for software project planning and estimating, the software project manager and the software test manager develop a baseline resource allocation and schedule. The software project manager and the software test manager then negotiate with both the project manager and line management to identify and secure the needed resources. All resources which are needed by the various software development phases in this methodology should be identified. These include software developers (software analysts, software designers, and programmers), test engineers, equipment, and budget. In addition to software development and testing personnel, domain experts are identified and allocated. Domain experts include software surety experts, quality assurance personnel, human factors personnel, independent consultants, and personnel representing interfaces for concurrently developed software, existing software, COTS software, hardware and external systems. The selection of a surety expert or surety experts with software experience is a requisite of this task.</p> <p>In addition to resources, specific training needs and alternatives to satisfying those needs are identified.</p>

## Project Planning Phase

<b>Exit Criteria</b>	Line management, the project manager, the software project manager, and the software test manager agree to the allocated resources and baseline schedule. Training needs are documented and planned.
<b>Outputs</b>	<p>The outputs from this task are:</p> <ul style="list-style-type: none"><li>• the allocated resources for the project,</li><li>• the baseline schedule, and</li><li>• documentation of training needs and plans.</li></ul>
<b>References</b>	<ul style="list-style-type: none"><li>• Process Definition for Software Project Planning and Estimating [18]</li></ul>

## Project Planning Phase

### **Task 7: Establish Communications with Domain Experts**

<b>Objectives</b>	The purpose of this task is to establish communication with the domain experts (particularly the surety experts) who will be providing input to or support for the project. The allocated domain experts are involved in all phases of the project life cycle.
<b>Dependencies</b>	This task is dependent on Task 6.
<b>Responsibilities</b>	The software project manager is responsible for completion of this task.
<b>Inputs</b>	List of domain experts identified in Task 6.
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Contact with the domain experts is established and a communication process initiated. Provide project-specific information to each of the domain experts for their review. Discuss expectations and schedule with domain experts.
<b>Exit Criteria</b>	All appropriate domain experts have been contacted and a communication process has been established.
<b>Outputs</b>	None.
<b>References</b>	None.

**Task 8: Review High-Level Controlling Documents for the SDP and MTP**

<b>Objectives</b>	<p>The purpose of this task is:</p> <ul style="list-style-type: none"><li>• to review all possible controlling documents for their impacts on the SDP and MTP and</li><li>• to document the resulting impacts for later incorporation into these documents.</li></ul>
<b>Dependencies</b>	<p>This task is dependent on Task 5.</p>
<b>Responsibilities</b>	<p>The software project manager and the software test manager are responsible for completion of this task.</p>
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• PHA</li><li>• SOW</li></ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"><li>• Lessons Learned from Past Projects</li><li>• RMP</li><li>• SCMP</li><li>• SMP</li><li>• Software Standards and Processes</li><li>• SQAP</li><li>• System Specifications</li><li>• System Surety Requirements</li><li>• System-Level Project Plan and Schedule</li><li>• WBS</li></ul>
<b>Entrance Criteria</b>	<p>The inputs must be available before this task can start.</p>
<b>Task Description</b>	<p>The software project manager and software test manager review all relevant controlling documents to determine their impact on the SDP and the MTP. Controlling documentation must include the SOW and the PHA. Other controlling documents that might be available are listed above as inputs to this task. Using the PHA and the system surety requirements (if available), evaluate the impacts of all surety-critical requirements on the SDP and the MTP. If an SMP, an RMP, an SCMP, and/or an SQAP are available, determine their impacts on the SDP and MTP and how they are tailored for the project. If system specifications, a WBS, software standards/processes, and/or a system-level project plan and schedule exist, evaluate their impacts on the SDP and MTP. Document the resulting impacts for later incorporation into the SDP and MTP.</p>
<b>Exit Criteria</b>	<p>Analysis of the impacts of all controlling documentation is completed and documented. This information will later be incorporated into the SDP and MTP.</p>
<b>Outputs</b>	<p>The outputs from this task are the impacts of each controlling document on the content of the SDP and MTP.</p>
<b>References</b>	<p>None.</p>

**Task 9: Select Requirements Tracing Method and Tool(s)**

<b>Objectives</b>	The purpose of this task is: <ul style="list-style-type: none"><li>• to select a requirements tracing method and</li><li>• to evaluate and select a requirements tracing tool.</li></ul>
<b>Dependencies</b>	This task is dependent on Task 5.
<b>Responsibilities</b>	The software project manager and software test manager are responsible for completion of this task.
<b>Inputs</b>	The input to this task is: <ul style="list-style-type: none"><li>• SOW</li></ul> In addition, the following inputs may be included if appropriate: <ul style="list-style-type: none"><li>• SMP</li><li>• Software Standards and Processes</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	The software project manager and the software test manager review possible requirements tracing methods and select or modify an existing method. Samples of software requirements tracing tools are obtained for execution and analysis, or vendors are requested to demonstrate their tool. Through the use of software samples or through demonstrations, each software requirements tracing tool is evaluated based on the following criteria: <ul style="list-style-type: none"><li>• Does the tool allow tracing of requirements to design, design to code, and from requirements to test?</li><li>• Does the tool suit the needs of the project? In particular, does it support the selected requirements tracing method?</li><li>• Is the tool easy to use?</li><li>• Is the tool's documentation complete and easy to understand?</li><li>• Is there training available, and if so how much training time is needed?</li></ul> Other factors to consider are the tool's cost and the risk factors associated with the use of the tool. After the tools are evaluated, one is selected for use by the project for creating and maintaining the Requirements Trace Matrix (RTM).
<b>Exit Criteria</b>	Selection of a requirements tracing method is complete and evaluation of tracing tools is complete and documented.
<b>Outputs</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• requirements tracing method and</li><li>• requirements tracing tool.</li></ul>
<b>References</b>	<ul style="list-style-type: none"><li>• RTM™ Tool[23]</li><li>• Requisite™ Tool[24]</li></ul>

**Task 10: Select Document Templates**

<b>Objectives</b>	The purpose of this task is: <ul style="list-style-type: none"><li>• to modify and approve the SSTP template from [1] ,</li><li>• to select and approve an MTP template,</li><li>• to select and approve an SDD template,</li><li>• to select and approve an SDP template,</li><li>• to select and approve an SRS template,</li><li>• to select and approve an SCMP template if necessary, and</li><li>• to select and approve an SQAP template if necessary.</li></ul>
<b>Dependencies</b>	This task is dependent on Task 6.
<b>Responsibilities</b>	The software project manager, software development team, the software test manager, and the testing team are responsible for completion of this task.
<b>Inputs</b>	The inputs to this task are: <ul style="list-style-type: none"><li>• Available SDP Templates, MTP Templates, SRS Templates, SDD Templates, and the SSTP Template from the Preferred Processes for SW Development [1].</li><li>• SOW</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	The software project manager along with the software development team members review available templates and make a selection for the SDP, SRS, SDD, and, if needed, the SCMP and SQAP. The software test manager along with the test team members identify and select a template for the MTP and modify the SSTP template if needed.
<b>Exit Criteria</b>	The selected templates are approved by appropriate software development and testing personnel.
<b>Outputs</b>	The outputs from this task are the selected templates for the project documentation.
<b>References</b>	<ul style="list-style-type: none"><li>• IEEE Std. 1058.1-1987 [9]</li><li>• IEEE Std. 1228-1994 [10]</li><li>• IEEE Std. 730-1989 [4]</li><li>• IEEE Std. 828-1990 [5]</li><li>• MIL-STD-498 [2]</li><li>• Sandia Preferred Processes for Software Development [1]</li><li>• Sandia Software Guidelines, Volume 2, Documentation [13]</li></ul>

**Task 11: Develop Outline/Framework for SDP and MTP**

<b>Objectives</b>	<p>The purpose of this task is to complete initial versions of the SDP and the MTP. These versions contain an outline of the SDP and MTP with some of the introductory sections completed. Initial customer feedback may be obtained.</p> <p>The purpose of the SDP is to establish realistic plans for performing the software engineering activities and for managing and tracking the software project. The purpose of the MTP is to guide and control all testing efforts and to define the overall test strategy and high-level test plans.</p>
<b>Dependencies</b>	<p>This task is dependent on Tasks 7, 8, 9 and 10.</p>
<b>Responsibilities</b>	<p>The software project manager and software test manager are responsible for completion of this task.</p>
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• Allocated resources for the project from Task 6</li><li>• Baseline schedule from Task 6</li><li>• Documentation of training needs and plans from Task 6</li><li>• Formal documentation of project management requirements from Task 4</li><li>• Formal statement of the project's scope from Task 3</li><li>• Impacts of each controlling document on the content of the SDP and MTP from Task 8</li><li>• Project's software development objectives from Task 2</li><li>• Project's surety objectives from Task 2</li><li>• Project's surety testing objectives from Task 2</li><li>• Project's testing objectives from Task 2</li><li>• Selected MTP Template from Task 10</li><li>• Selected SDP Template from Task 10</li><li>• SOW</li></ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"><li>• System Specifications</li><li>• System Surety Requirements</li></ul>
<b>Entrance Criteria</b>	<p>The inputs must be available before this task can start.</p>

## Project Planning Phase

<b>Task Description</b>	<p>Execution of this task requires completion of the initial versions of the SDP and MTP and, if required, a successful meeting with the customer to discuss the draft documents. The following are examples of the introductory sections usually documented in an SDP: scope, objectives, system overview, document overview, references, project organization, and project resources.</p> <p>The following are examples of the introductory sections usually documented in an MTP: scope, objectives, references, resource and schedule constraints, test planning philosophy, and testing resources.</p> <p>The completion of this task depends on the templates chosen for the SDP and the MTP, i.e., IEEE, MIL-STD. Refine tasks and milestone schedules for both software development and testing based on information gathered in previous tasks. Coordinate the top-level schedules with management and with the customer.</p> <p>The selected SDP and MTP outlines are customized to include sections that describe all surety-related processes and activities necessary to produce highly-reliable, safe, and/or secure software. Begin defining processes for verification of surety attributes using the surety testing objectives from Task 2. Possible verification activities are analysis of the product and testing of the product's attributes (safety, security, etc.). A stand-alone Software Safety Plan may be developed which identifies milestones for surety activities called out in subsequent phases of the software life cycle.</p>
<b>Exit Criteria</b>	<p>The initial versions of the SDP and MTP are completed by the software project manager and the software test manager and approved by appropriate software development and testing personnel. A successful meeting obtaining customer approval may also take place.</p>
<b>Outputs</b>	<p>The outputs from this task are:</p> <ul style="list-style-type: none"><li>• initial version of the SDP,</li><li>• initial version of the MTP and,</li><li>• if appropriate, an approval memo from the customer.</li></ul>
<b>References</b>	<ul style="list-style-type: none"><li>• IEEE Std. 1058.1-1987 [9]</li><li>• MIL-STD-498 [2]</li><li>• IEEE Std. 730-1989 [4]</li><li>• IEEE Std. 828-1990 [5]</li><li>• IEEE Std. 1228-1994 [10]</li><li>• Sandia Software Guidelines, Volume 2, Documentation [13]</li><li>• Sandia Preferred Processes for Software Development [1]</li></ul>

## Project Planning Phase

### **Task 12: Develop SDP and MTP**

<b>Objectives</b>	The purpose of this task is to develop draft versions of the SDP and the MTP. These draft versions have all sections completed but may have TBDs.
<b>Dependencies</b>	This task is dependent on Task 11.
<b>Responsibilities</b>	The software project manager and software test manager are responsible for completion of this task.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• Allocated resources for the project from Task 6</li><li>• Baseline schedule from Task 6</li><li>• Documentation of training needs and plans from Task 6</li><li>• Formal documentation of project management requirements from Task 4</li><li>• Formal statement of the project's scope from Task 3</li><li>• Impacts of each controlling document on the content of the SDP and MTP from Task 8</li><li>• Initial versions of the SDP and MTP from Task 11</li><li>• Project's software development objectives from Task 2</li><li>• Project's surety objectives from Task 2</li><li>• Project's testing objectives from Task 2</li><li>• Project's surety testing objectives from Task 2</li><li>• SOW</li></ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"><li>• System Specifications</li><li>• System Surety Requirements</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>Execution of this task involves drafting all sections of the SDP and the MTP. If a section is not applicable, it is noted. In particular for the SDP, the management approach for the project is defined, the technical approach for the project is formulated, and the project estimates are documented. In addition, the plans for configuration management, software quality assurance, and risk management are documented as sections or appendices in the SDP or as separate documents. The outputs from Task 2, Task 3, Task 4, Task 6, and Task 8 are incorporated where appropriate. Sections relevant to testing are coordinated with the MTP. At the end of this task and prior to inspection, the draft version of the SDP is placed under configuration management according to the Software Configuration Management Plan (SCMP).</p> <p>For the MTP, the testing approach and strategies are defined and testing estimates are documented. The outputs from Task 2, Task 3, Task 4, Task 6, and Task 8 are incorporated where appropriate. Relevant sections are coordinated with the SDP. At the end of this task and prior to inspection, the draft version of the MTP is placed under configuration management.</p> <p>All sections that describe the project-specific surety-related processes and activities are completed and informally reviewed by the surety experts.</p>

## Project Planning Phase

### Exit Criteria

The preliminary versions of the SDP and the MTP are completed and all surety related sections have been informally reviewed by surety experts. In addition, the software project manager and software test manager verify that the SDP and the MTP are ready for inspection and have been placed under configuration management according to the CM section in the SCMP.

### Outputs

The outputs from this task are:

- preliminary version of the SDP,
- preliminary version of the MTP,
- preliminary version of the SQAP, if appropriate, and
- preliminary version of the SCMP, if appropriate.

### References

- IEEE Std. 730-1989 [4]
- IEEE Std. 828-1990 [5]
- IEEE Std. 1058.1-1987 [9]
- IEEE Std. 1228-1994 [10]
- MIL-STD-498 [2]
- Sandia Preferred Processes for Software Development [1]
- Sandia Software Guidelines, Volume 2, Documentation [13]

**Task 13: Inspect SDP and MTP**

The Inspection Process is described in detail in Section 5.1 of this document. The following paragraphs describe the particulars of inspecting the SDP and the MTP. Note that the SDP and MTP are inspected separately with different review teams.

**Inspection Materials**      SDP, MTP

**Task Dependencies**      This task is dependent on Task 12.

**Support Materials**      The following materials may be supplied to the inspection team as supporting material for the inspection of the SDP:

- PHA
- SOW
- SMP (if available)
- System Specifications (if available)
- System Surety Requirements (if available)

The following materials may be supplied to the inspection team as supporting material for the inspection of the MTP:

- PHA
- SOW
- SMP (if available)
- Higher level test and evaluation plan (if available)
- System Specifications (if available)
- System Surety Requirements (if available)
- Testing Requirements from any verification agencies

**Inspection Team**      The SDP inspection team includes the surety expert, the software test manager, the software project manager, and a member from each development team (requirements, design, implementation, integration, and test).

The MTP inspection team includes the surety expert, the software project manager, the software test manager, all test engineers assigned to the project, and, as required, any members from the development teams.

**General Objectives**      The SDP inspection team is responsible for verifying that the SDP reflects the project objectives and scope; identifies and mitigates project risks; and adequately estimates the project resources, costs, and schedule. The document is also examined for consistency, completeness, and comprehensiveness.

The MTP inspection team is responsible for verifying that the MTP reflects the project's testing objectives and scope; identifies all software testing issues, and adequately estimates the testing resources, costs, and schedule. The document is also examined for consistency, completeness, and comprehensiveness.

**Testing Objectives**      The software test manager is primarily responsible for ensuring that the test plans (if any) documented in the SDP are consistent with the MTP.

**Surety Objectives**      The surety expert is responsible for ensuring that the project-specific surety-related processes and activities are described accurately and completely in the SDP and the MTP.

## Project Planning Phase

### **Outputs**

The outputs from this task are:

- the SDP inspection reports and the revised SDP and
- the MTP inspection reports and the revised MTP.

**Task 14: Baseline and Distribute SDP and MTP**

<b>Objectives</b>	The purpose of this task is to allow the customer to formally approve the final SDP and the final MTP, and distribute these documents to all appropriate project personnel.
<b>Dependencies</b>	This task is dependent on Task 13.
<b>Responsibilities</b>	The software project manager and the software test manager are responsible for completion of this task.
<b>Inputs</b>	The inputs to this task are the final versions of the SDP and the MTP as revised per Task 13.
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>Execution of this task requires that the SDP be signed off by the software project manager, the customer, the surety expert, the software test manager, and a member from each development team (requirements, design, implementation, integration, and test). The final SDP is then distributed to all appropriate project team members as well as the customer.</p> <p>In addition, this task requires that the MTP be signed off by the software test manager, the customer, the surety expert, the software project manager, members from each development team, and each member of the testing team. The final MTP is then distributed to all appropriate project team members as well as the customer.</p>
<b>Exit Criteria</b>	The SDP and the MTP have been distributed to all appropriate personnel.
<b>Outputs</b>	<p>The outputs from this task are:</p> <ul style="list-style-type: none"><li>• the SDP and</li><li>• the MTP.</li></ul> <p>The SDP and the MTP are intended to be living documents throughout the project life cycle and are updated frequently to reflect the current state of the project. The SDP and the MTP are intended to be used as effective software and testing management tools.</p>
<b>References</b>	None.

## 3.2 Requirements Phase

### 3.2.1 Overview

The Requirements Phase takes a high-level product description and often ambiguous and untestable customer requirements and outputs a Software Requirements Specification (SRS). In addition to the SRS, testing activities are performed in preparation for testing activities executed in later phases. These testing activities are documented in the Software System Test Plan (SSTP).

The Requirements Phase provides input primarily in the form of an SRS and an SSTP to the following phases:

- Design
- Implementation
- Integration
- Test

The SRS is a contract between the customer and the developers and testers. Other phases use the SRS as the fundamental requirement guide. The Requirements Phase does not include the definition or specification of internal system constraints, limitations, or design guidelines. Specifications of this nature are considered design decisions and are included in the SDD. The requirements specification should describe *what* will be developed, not *how* it will be developed.

### 3.2.2 Inputs and Suppliers

The inputs utilized in the Requirements Phase are summarized in Table 3-5.

**Table 3-5 - Requirements Phase Inputs and Suppliers**

Input	Supplier
Customer's Existing Environment	Customer Project Planning Phase
Modification Request	Various Sources
MTP	Project Planning Phase
PHA	Customer System Engineers Surety Experts
SCMP, SQAP	Project Planning Phase
SDP	Project Planning Phase
SOW	Customer
SRS Template	Project Planning Phase
SSTP Template	Project Planning Phase
Software Standards and Processes	Organization Processes, Department Processes
System Specifications	System Engineers, Customer

### 3.2.3 Outputs and Customers

Table 3-6 contains the outputs from this phase.

## Requirements Phase

**Table 3-6 - Requirements Phase Outputs and Customers**

<b>Output</b>	<b>Customer</b>
Software Hazard Analysis (SHA)	Design Team
Surety Folder	Design Team
SRS	Design Phase
SSTP	Implementation Phase
RTM	Integration Phase
	Test Phase
	Quality Assurance
MTP	Design Phase
Metrics	Quality Assurance, Other Projects, Software Project Manager
Modification Requests	Various Phases

### 3.2.4 Metrics

This phase's metrics should provide a measure of the effectiveness of the Requirements Phase and include the following:

- requirements stability metrics (number of requirements added, deleted, or modified) gathered on a periodic basis and
- SRS inspection metrics as reported on the Inspection Summary form.

The Requirements Phase tasks are summarized in the Table 3-7. Each of these tasks is described in greater detail in Section 3.2.7.

**Table 3-7 - Requirements Phase Task Summary**

<b>Task</b>	<b>Description</b>	<b>Dependencies</b>
1	Define and Schedule SRS Activities	
2	Write SRS Overview and Get Customer Feedback	1
3	Perform a Software Hazard Analysis (SHA)	2,4
4	Prepare Draft SRS	2, 3
5	Evaluate Test Support Tools	4
6	Inspect SRS	4
7	Get Customer Feedback	6
8	Prepare Planning Section of SSTP	6
9	Approve and Distribute SRS	7

# Requirements Phase

## 3.2.5 Task Flow

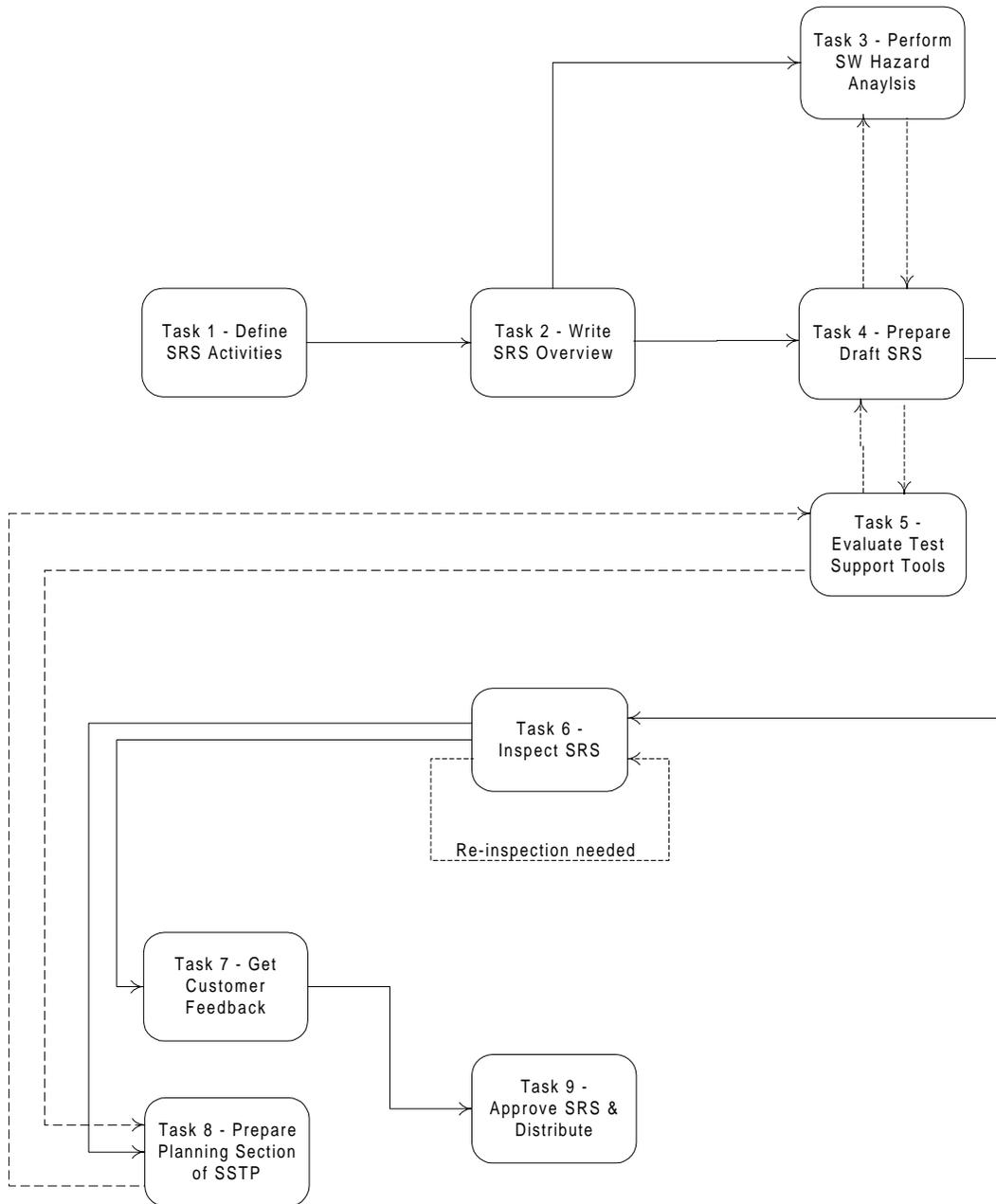


Figure 3-2 Requirements Phase Task Flow

### **3.2.6 Modification Requests**

#### **3.2.6.1 Generation of Requests**

Tasks within the Requirements Phase may generate modification requests that can affect the outputs of other phases. For example, modification requests may be submitted to the Project Planning Phase if the schedule or resources allocated to the project are judged inadequate to accomplish the desired requirements. Risks associated with system hazards identified in Task 3 may impact the scope of the project. Software functions could be reallocated to hardware and affect schedule and resources.

#### **3.2.6.2 Receipt and Processing of Requests**

The Requirements Phase may receive modification requests from other phases to update an output from the Requirements Phase. These modification requests are handled by revisiting appropriate tasks from the Requirements Phase and revising those tasks' outputs. When tasks are revisited, the time devoted to each task depends on the scope of the modification request. Tasks which may have taken weeks to perform during the initial completion of the Requirements Phase may take only hours to perform when addressing a modification request. If a modification request is received during the initial performance of the Requirements Phase, only those tasks that have been completed need to be revisited. Tasks in progress or not yet begun are performed as usual with the new information from the modification request added as an input to the task.

If a change is requested to the SRS, Tasks 3 through 6 and 8 are performed. In Task 3, the surety expert should revise the SHA if any additional software functions and/or hazards are identified related to the modification request. Task 5 may need to be revisited if additional or modified test support tools are required by new software functionality. Task 6 is performed with the scope of the inspection reduced to the changes made to the SRS. If the changes are shown to be minor, then an informal review may be performed instead of a formal inspection at the discretion of the software project manager. If Design Phase tasks have begun, then a modification request is submitted to the Design Phase to propagate the changes to the SRS into the software design.

If a change is requested to the planning section of the SSTP, only Task 8 is revisited.

### 3.2.7 Detailed Task Descriptions

This section provides the detailed description of each task listed in Table 3-7. Each task should be completed to successfully carry out the Requirements Phase of development.

#### **Task 1: Define and Schedule SRS Activities**

<b>Objectives</b>	The purpose of this task is to plan tasks associated with the generation of an SRS.
<b>Dependencies</b>	None.
<b>Responsibilities</b>	The SRS author is responsible for managing all tasks required for the completion of Task 1.
<b>Inputs</b>	The input to this task is the SDP.
<b>Entrance Criteria</b>	The inputs must be available before this task can start. The software project manager must authorize the SRS author to proceed.
<b>Task Description</b>	Execution of this task requires the refinement of the schedule in the SDP for all tasks included in the Requirements Phase.
<b>Exit Criteria</b>	The software project manager reviews schedule milestones and must approve the SRS schedule.
<b>Outputs</b>	The output from this task is a completed and approved schedule for all tasks included in the Requirements Phase.
<b>References</b>	None.

## Requirements Phase

### **Task 2: Write SRS Overview and Get Customer Feedback**

<b>Objectives</b>	The purpose of this task is to complete a draft version of the SRS overview and obtain initial customer feedback.
<b>Dependencies</b>	This task is dependent upon Task 1.
<b>Responsibilities</b>	The SRS author is responsible for managing all activities required for the completion of this task.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• PHA</li><li>• SRS Template</li></ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"><li>• Customer Specification</li><li>• System Specifications</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>To initiate this task, the inputs identified above are carefully reviewed. An objective of this review is to identify specific domain experts, which are allocated in the Project Planning Phase, necessary for the formulation of requirements for the software. Also, questions concerning lack of clarity in the inputs to this process or any conflicting information and/or requirements should be documented.</p> <p>Meet with the domain experts to discuss and resolve the issues identified in the review. Formally document decisions and agreements made during these meetings and distribute to all participants for review. These meeting minutes will serve as an additional input to this task and Task 3.</p> <p>If domain experts are unavailable or questions cannot be resolved, consider prototyping as a method for understanding and/or eliciting requirements.</p> <p>Execution of this task requires completion of a draft version of the overview of the SRS, project management approval for customer contact, and a successful meeting with the customer to discuss the draft. Note that customer contact is not a gating-function for exiting this task. The objective of customer contact is the customer and the SRS author reach a shared general understanding of the proposed product. The SRS author should make every reasonable attempt to contact the customer to clarify and confirm the definition and description of the software product. However, customer contact is not always feasible or advised; in these cases work should continue without customer feedback.</p> <p>Note that customer contact is not limited to only Task 2 and Task 7. In some cases, working meetings occur during many of the Requirements Phase tasks. The fundamental reason for identifying the initial and second customer feedback tasks is to acknowledge the need for resolution and clarification of requirements for some software products.</p>
<b>Exit Criteria</b>	The overview of the SRS is completed and the project manager and software project manager review it. The SRS is presented to the customer when possible.
<b>Outputs</b>	<p>The outputs from this task are:</p> <ul style="list-style-type: none"><li>• a draft version of the overview of the SRS and</li><li>• if appropriate, an approval memo from the customer.</li></ul>
<b>References</b>	None.

**Task 3: Perform a Software Hazard Analysis (SHA)**

<b>Objectives</b>	The purpose of this task is to identify and document potential hazards associated with the functionality of the software in the system.
<b>Dependencies</b>	This task is dependent on Task 2 and interdependent on Task 4.
<b>Responsibilities</b>	The SRS author is responsible for providing all inputs required for completion of this task. A surety expert is responsible for performing an independent SHA based on the SRS Overview. Domain knowledge experts as well as the test engineer should be available for review of the SHA.
<b>Inputs</b>	The inputs to this task are: <ul style="list-style-type: none"><li>• SRS Overview from Task 2</li><li>• PHA</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	All functions performed by the software are identified and defined. Each software function is then examined to determine if that function has any impact on system surety. Hazards associated with software functions are defined explicitly and ranked based on the possible impact on the system's surety (criticality). This task may be performed iteratively as more information is incorporated into the SRS. The list of hazards should be incorporated into an initial surety folder by the surety expert. For more information on the contents of a surety folder, see Appendix D.
<b>Exit Criteria</b>	The SHA is complete, and the SRS author, the software project manager, and domain experts review it. The SHA is presented to the customer when possible.
<b>Output</b>	The outputs are an SHA and an initial surety folder that have been reviewed and approved by the SRS author.
<b>References</b>	<ul style="list-style-type: none"><li>• MIL-STD-882B [3]</li><li>• IEEE 1228 (provides some general guidance about SHA) [10]</li><li>• Software Systems Safety Handbook [26]</li><li>• A Study on Hazard Analysis in High Integrity Software Standards and Guidelines [27]</li><li>• Requirements for the Analysis of Safety Critical Hazards [28]</li><li>• NASA Software Safety Standard [29]</li></ul>

## Requirements Phase

### **Task 4: Prepare Draft SRS**

<b>Objectives</b>	The purpose of this task is to complete the draft version of the SRS for inspection.
<b>Dependencies</b>	This task is dependent on Task 2 and interdependent on Task 3.
<b>Responsibilities</b>	The SRS author is responsible for managing all activities required for the completion of this task. The test engineer reviews all requirements for testability.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• SRS Overview from Task 2</li><li>• Staffing of the SRS team from the SDP</li><li>• Interim SHA from Task 3</li></ul> <p>In addition, the following input may be included if appropriate:</p> <ul style="list-style-type: none"><li>• SOW</li><li>• System Specifications</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>Execution of this task requires formation of a team of experienced developers, requirements analysts, and surety experts made available from identified project resources. This team is the SRS team. The purpose of forming this team is to provide an understanding of the general functional areas within development and areas that are critically important for the specification of the software product. The SRS author should also become familiar with interactions, interfaces and sources of requirements and gain a grasp of the scope and implications of each requirement. The requirements may be described in the SRS with diagrams such as data flow diagrams and state transition diagrams or formal notation. Sections should be added to the SRS wherever appropriate to completely specify the product.</p> <p>A Requirements Trace Matrix (RTM) should be developed and added to the SRS. The RTM will allow each requirement to be traced through Design, Implementation, Integration, and Test Phases. The initial RTM included in the SRS should trace requirements back to system-level requirements documented in the system specifications, if available.</p> <p>For HCSs, the surety requirements must be identified, preferably in an appendix or a separate document, from the non-surety-related requirements. This separation allows an additional emphasis on formulation and tracking of the surety requirements.</p> <p>A Surety Requirements Trace Matrix that relates surety requirements to hazards identified in the SHA should be developed as a tool for brainstorming, evaluating, and tracking surety requirements. Each hazard should have at least one surety requirement identified that mitigates or controls that hazard. This matrix is maintained throughout the development process to allow surety requirements to be traced from a source, example: customer-requested requirement though design to code and test. The matrix is added to the surety folder.</p> <p>Formal methods, specifically formal specifications, may be used to define surety-critical requirements. Formal mathematical notation of requirements is a less ambiguous and easier analyzed representation of requirements than natural language representations. Formal methods allow a validation of a behavior model of the requirements by</p>

## Requirements Phase

mathematical proof. Formal methods are further described in Appendix C.

The test engineer is responsible for reviewing all requirements in the SRS for testability and participating in the inspection of the SRS. Ideally, test cases from Task 2 in the Design Phase would be developed at this time.

### **Exit Criteria**

The SRS team must be formed and a draft version of the SRS and a Surety Requirements Trace Matrix must be completed. All issues identified in SHA have been addressed. The SRS is placed under configuration management according to the SCMP.

### **Outputs**

The outputs from this task are:

- a draft version of the SRS,
- a Surety Requirements Trace Matrix,
- the RTM, and
- an updated surety folder.

### **References**

None.

## Requirements Phase

### **Task 5: Evaluate Test Support Tools**

<b>Objective</b>	The purpose of this task is to select and initiate procurement of test tools to support the testing activities in the Implementation, Integration, and Test Phases.
<b>Dependencies</b>	This task is interdependent on Task 4 and Task 8.
<b>Responsibilities</b>	The test engineer is responsible for assessing the test environment and tool needs for the project.
<b>Inputs</b>	The inputs to this task are: <ul style="list-style-type: none"><li>• MTP</li><li>• Interim SRS</li><li>• Plan for software system test effort from Task 8</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	The test environment and test tool needs are assessed for unit (Implementation Phase), Integration (Integration Phase), and system (Test Phase) testing. The test engineer researches the availability of test tools to accomplish the objectives of the testing as specified in the MTP. Tools are selected and procurement is initiated, if necessary. Any staff training necessary to utilize the tools for testing is also identified. The test engineer should focus special attention on surety requirements. Surety requirements may require special tools since an undesired system state may need to be simulated. Sections describing test environment and tools in the MTP should be updated.
<b>Exit Criteria</b>	The test environment and test tool sections in the MTP must be updated and approved by the software test manager.
<b>Outputs</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• an updated MTP and</li><li>• procurement of tools needed to support the Implementation, Integration, and Test Phases.</li></ul>
<b>References</b>	None.

## Requirements Phase

### **Task 6: Inspect SRS**

The Inspection Process is described in detail in Section 5.1 of this document. The following paragraphs describe the particulars of inspecting the SRS.

**Inspection Materials** SRS, RTM

**Task Dependencies** This task is dependent on Task 4.

**Support Materials** The following materials may be supplied to the inspection team as supporting material for the inspection:

- SHA
- SOW
- Surety Folder
- System Specifications

**Inspection Team** The inspection team should include the SRS author, domain experts, surety experts, test engineer, and some design team members

**General Objectives** The inspection team members are responsible for verifying the completeness and correctness of the SRS and that the SRS conforms to the SRS template identified in the Project Planning Phase.

**Testing Objectives** The test engineer assesses the draft requirements to determine if each requirement is testable. A testable requirement has the following attributes: clearly defined objectives, unambiguous specifications, and sufficient detail to always be able to define expected results.

**Surety Objectives** The surety expert should confirm that all hazards identified in the SHA have specific requirements that mitigate or control each hazard. If a hazard can not be controlled or mitigated, the rationale for accepting the associated risk is documented in the surety folder. The surety folder is updated to capture any changes to the surety requirements made during the Inspection Process.

**Outputs** The outputs of the inspection are:

- the inspection reports,
- revised SRS, and
- updated surety folder.

## Requirements Phase

### **Task 7: Get Customer Feedback**

<b>Objectives</b>	The purpose of this task is to review software requirements with the customer using the SRS.
<b>Dependencies</b>	This task is dependent on Task 6.
<b>Responsibilities</b>	The SRS author is responsible for managing all activities required for the completion of this task.
<b>Inputs</b>	The input for this task is the inspected SRS.
<b>Entrance Criteria</b>	The inputs must be available before this task can start. Project management must have approved customer contact.
<b>Task Description</b>	The SDP determines whether this task is applicable to the project. Execution of this task includes preparing an SRS for the customer, meeting with the customer, resolving any issues and modifying the SRS. The formality of this review is determined by the SDP.
<b>Exit Criteria</b>	The exit criteria for this task requires receipt of customer approval of the SRS. This approval may take the form of a memo from the customer. If the customer requests only minor changes, continue with Task 8. If the customer requests major changes, return to Task 4.
<b>Outputs</b>	The output from this task is a customer approved SRS.
<b>References</b>	None.

**Task 8: Prepare Planning Section of Software System Test Plan**

<b>Objective</b>	The purpose of this task is to plan the software system test effort and document it in the SSTP.
<b>Dependencies</b>	This task is interdependent on Task 5 and dependent on Task 6.
<b>Responsibilities</b>	The test engineer is responsible for planning and documenting the software system test effort in the SSTP. This does not include the actual test cases and procedures.
<b>Inputs</b>	The inputs to this task are: <ul style="list-style-type: none"><li>• Inspected SRS</li><li>• MTP</li><li>• SHA</li><li>• SSTP template</li><li>• Selected test support tools from Task 5</li><li>• Surety Folder</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Based on the inspected SRS and the surety information contained in both the SHA and the Surety Folder, the test engineer prepares a plan for the software system test effort and documents it in the planning sections of the SSTP. These sections identify: test items, test approach, test environment, pass/fail criteria, suspension/resumption criteria, deliverables, tasks, test responsibilities, resources, and schedule. The test cases and test procedures are not included in this version of the SSTP.
<b>Exit Criteria</b>	The planning sections of the SSTP is complete, and the software test manager reviews the SSTP.
<b>Outputs</b>	The output from this task is a draft SSTP.
<b>References</b>	None.

## Requirements Phase

### **Task 9: Approve and Distribute SRS**

<b>Objectives</b>	The purpose of this task is to obtain formal approval of the SRS.
<b>Dependencies</b>	This task is dependent on Task 7.
<b>Responsibilities</b>	The SRS author is responsible for managing all activities required for the completion of this task.
<b>Inputs</b>	The input for this task is the customer approved SRS.
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Execution of this task requires that the SRS author obtain the required signatures as defined in the SDP. The author must then distribute a copy of the SRS to internal customers.
<b>Exit Criteria</b>	The SRS author obtains all required signatures. The SRS author must notify the software project manager that the SRS has been distributed.
<b>Outputs</b>	The output from this task is the SRS.
<b>References</b>	None.

### 3.3 Design Phase

#### 3.3.1 Overview

The purpose of the Design Phase is to produce a blueprint for the implementation of the software product. In addition, testing activities are performed in this phase which are preparatory to testing which is performed in later phases. This phase takes as input the SRS, SHA, the interim SSTP, and any related system specifications that exist.

The Design Phase provides input primarily in the form of a Software Design Description (SDD), a Software System Test Plan (SSTP), and an Integration Test Plan to the following phases:

- Implementation
- Integration

The SSTP is expanded in the Integration Phase to include detailed test procedures. These procedures are derived from the test cases produced during the Design Phase. The Integration Test Plan, produced in the Design Phase, is executed as part of the Integration Phase.

The Implementation Phase uses the SDD as its fundamental guide. The design described in the SDD must satisfy all requirements stated in the SRS. Accordingly, implementation based on the SDD must also satisfy those requirements. The Design Phase should produce the definition or specification of internal system constraints, limitations and design guidelines. Specifications of this nature are considered design decisions. Other phases also provide input in the form of requests for modifications to the design. These requests document the need for modification to the existing SDD.

#### 3.3.2 Inputs and Suppliers

The inputs utilized in the Design Phase are summarized in Table 3-8:

**Table 3-8 - Design Phase Inputs and Suppliers**

<b>Input</b>	<b>Supplier</b>
Modification Requests	Various Sources
MTP	Requirements Phase
RTM	Requirements Phase
SCMP, SQAP	Project Planning Phase
SDD Template	Project Planning Phase
SDP	Project Planning Phase
SHA	Requirements Phase
SOW	Customer
SRS	Requirements Phase
SSTP	Requirements Phase
Surety Folder	Requirements Phase
System Specifications	System Engineers

#### 3.3.3 Outputs and Customers

Table 3-9 contains the outputs from this phase:

## Design Phase

**Table 3-9 - Design Phase Outputs and Customers**

<b>Output</b>	<b>Customer</b>
Development Tools	Implementation Phase
DHA	Implementation Phase
Existing Code (libraries, COTS)	Implementation Phase
Integration Test Plan	Integration Phase
Metrics	Quality Assurance, Software Project Manager, Other Projects
Modification Requests	Various Phases
RTM	Implementation Phase
SDD	Implementation Phase
SSTP	Integration Phase
Surety Folder	Implementation Phase

### 3.3.4 Metrics

This phase's metrics should provide a measure of the effectiveness of the Design Phase and includes the SDD inspection metrics as reported on the Inspection Summary Form.

### 3.3.5 Task Summary

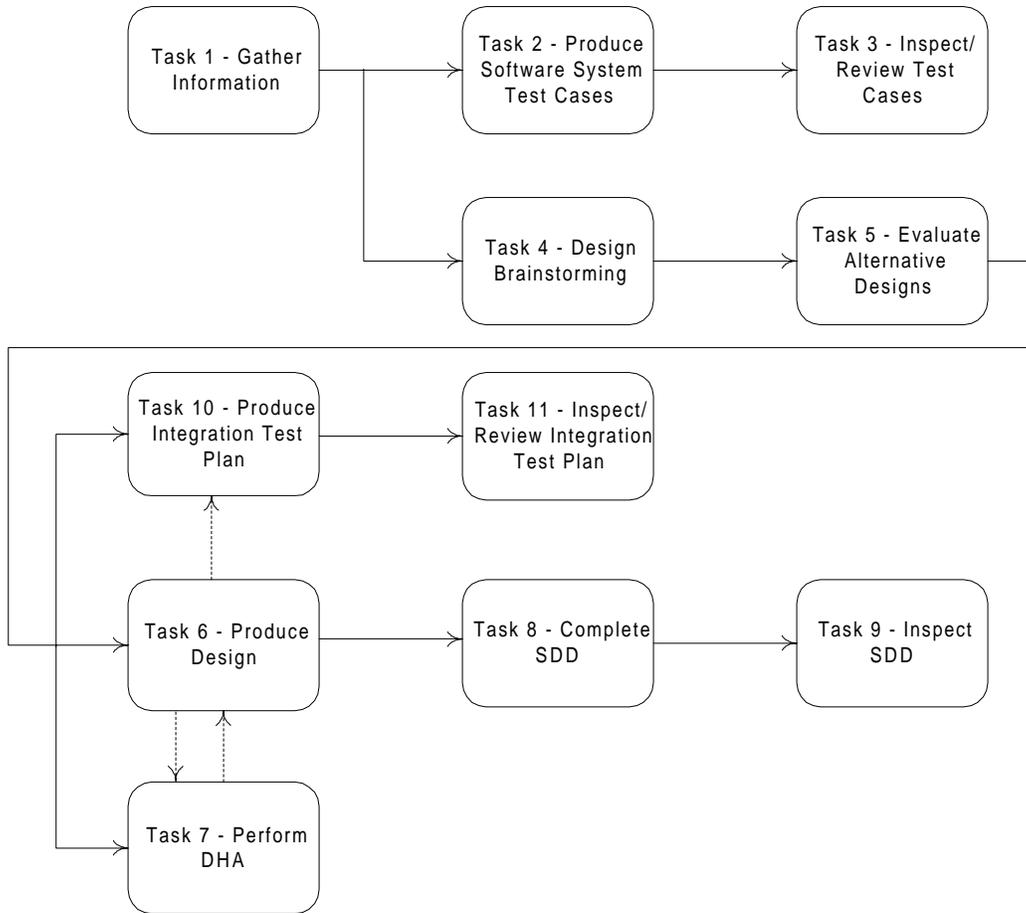
The Design Phase tasks are summarized in Table 3-10. Each of these tasks is described in greater detail in Section 3.3.7.

**Table 3-10 - Design Phase Task Summary**

<b>Task</b>	<b>Description</b>	<b>Dependencies</b>
1	Gather Information	
2	Produce Software System Test Cases	1
3	Inspect/Review Test Cases	2
4	Hold Design Brainstorming Meetings	1
5	Investigate and Evaluate Alternative Designs	4
6	Produce Design	5,7
7	Perform Design Hazard Analysis (DHA)	5,6
8	Complete SDD	6
9	Inspect SDD	8
10	Produce Integration Test Plan	5,6
11	Inspect/Review Integration Test Plan	10

## Design Phase

### 3.3.6 Task Flow



**Figure 3-3. Design Phase Task Flow**

### **3.3.7 Modification Requests**

#### **3.3.7.1 Generation of Requests**

There are several tasks performed during the Design Phase in which modification requests may be generated which affect the outputs of other phases. For instance, modification requests may be submitted to the Requirements Phase in order to modify the SRS. This may become necessary if during the production of software system test cases in Task 2, a requirement is found to be unclear or untestable, or if a requirement is found to be missing. Similarly, at any time during the production of the design in Tasks 4-6, a requirement could be found to be misstated or unachievable. During the performance of the Design Hazard Analysis (DHA) in Task 7, it could be found that the requirements on the product can not be met without compromising the surety of the system. Modification requests may also be submitted to the Project Planning Phase if changes are found to be necessary to the schedule or resource allocations specified in the SDP.

#### **3.3.7.2 Receipt and Processing of Requests**

The Design Phase may receive modification requests in order to update an output from the Design Phase. These modification requests are handled by revisiting appropriate tasks from the Design Phase. When tasks are revisited, the time devoted to each task will depend on the scope of the modification request. Tasks which may have taken weeks to perform during the initial completion of the Design Phase may take only hours to perform when addressing a modification request. If a modification request is received during the initial performance of the Design Phase, only those tasks which have been completed need to be revisited. Tasks in progress or not yet begun are performed as usual taking the new information from the modification request into account.

If a change is requested to the SDD, then Tasks 4-9 are performed. A brainstorming meeting is held amongst the developers in order to explore the impact that the modification request makes on the design. Possible solutions are explored, a final decision is made by the software project manager, and the SDD is updated. Task 7 is performed by having the surety expert review the changes to the SDD for possible impact on the DHA. Task 9 is performed with the scope of the inspection reduced to the changes made to the SDD. If the changes are minor, then an informal review may be performed instead of a formal inspection at the discretion of the software project manager. Depending on the scope of the change to the SDD, Tasks 10 and 11 may need to be revisited in order to update the integration tests. If work has already begun or been completed in the Implementation Phase, then a modification request will most likely need to be submitted so that the changes made to the SDD are propagated to the source code and unit tests.

If a change is requested to the software system test cases then Tasks 2 and 3 are revisited. If a change is requested to the integration test plan, then Tasks 10 and 11 are revisited. As with changes to the SDD, the scope of the review/inspection task may be reduced to that of the actual changes.

### 3.3.8 Detailed Task Descriptions

This section provides the detailed description of each task listed in Table 3-10. Each task should be completed to successfully carry out the Design Phase of development.

#### **Task 1: Gather Information**

<b>Objectives</b>	The purpose of this task is to obtain all the input documentation necessary to begin the Design Phase.
<b>Dependencies</b>	None.
<b>Responsibilities</b>	The design team leader is responsible for collecting the required documents and distributing them to the design team and the test team. The design and test team members become familiar with the documents.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• SHA</li><li>• SOW</li><li>• SRS</li></ul> <p>In addition, the following input may be included if appropriate:</p> <ul style="list-style-type: none"><li>• System Specifications</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start. The entrance criteria for this task also includes completion of the Requirements Phase and designation of the design team and test team.
<b>Task Description</b>	Completion of this task requires the design team leader to gather the required documents. The design and test team members read and understand the documents and study the hardware/software, software/software and user interfaces.
<b>Exit Criteria</b>	Design team members are familiar with all specifications to the level necessary to make design decisions. Test team members are familiar with all specifications to the level necessary to begin writing test cases.
<b>Outputs</b>	None.
<b>References</b>	None.

**Task 2: Produce Software System Test Cases**

<b>Objectives</b>	The purpose of this task is to produce software system test cases from the SRS and document them in the SSTP.
<b>Dependencies</b>	This task is dependent on Task 1.
<b>Responsibilities</b>	The software test manager is responsible for managing this task. The test engineers are responsible for producing and documenting the test cases.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• RTM</li><li>• SHA</li><li>• SOW</li><li>• SRS</li><li>• SSTP</li><li>• Surety Folder</li></ul> <p>In addition, the following input may be included if appropriate:</p> <ul style="list-style-type: none"><li>• System Specifications</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>The test team reviews the requirements in the SRS and determines the major functional areas of the software product. Once the major functional areas are defined, the requirements are partitioned and assigned to these areas. Each functional area is assigned to a test team member who is responsible for producing test cases from the allocated requirements.</p> <p>The pass/fail criteria for each requirement is determined and documented in the SSTP. The testing method for each requirement is then chosen and documented in the SSTP. The requirements are then grouped and test cases documented in the SSTP. A single test case covers one or more requirements. The SHA is used to target certain requirements for special attention. Additional test cases are written in order to rigorously exercise surety-critical areas.</p> <p>As the test cases are written, the RTM is updated to map the test cases to the requirements. The Surety Folder is also updated at this time to include the mapping from test cases to surety requirements.</p>
<b>Exit Criteria</b>	Test cases are documented in the SSTP for all requirements and the RTM and Surety Folder are updated.
<b>Outputs</b>	<p>The outputs from this task are:</p> <ul style="list-style-type: none"><li>• the SSTP with system level test cases,</li><li>• the RTM with test cases mapped to requirements, and</li><li>• the Surety Folder</li></ul>
<b>References</b>	None.

**Task 3: Review/Inspect Test Cases**

<b>Objectives</b>	The purpose of this task is to review and/or inspect the software system test cases.
<b>Dependencies</b>	This task is dependent on Task 2.
<b>Responsibilities</b>	The software test manager is responsible for the completion of this task. Test and software development team members participate in the review or inspection of the test cases. The surety expert reviews the test cases for adequate coverage of surety-critical areas.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• MTP</li><li>• RTM</li><li>• SOW</li><li>• SRS</li><li>• SSTP</li><li>• Surety Folder</li></ul> <p>In addition, the following input may be included if appropriate:</p> <ul style="list-style-type: none"><li>• System Specifications</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	The MTP specifies whether the software system test cases will be inspected, or if a less formal review is sufficient. In either case, the test team members review the test cases for completeness and accuracy. Software development team members verify the interpretation of the requirements by the test team and the feasibility of the test cases. The surety expert ensures that the test cases give adequate coverage of the surety requirements. All reviewers verify the accuracy of the RTM.
<b>Exit Criteria</b>	The software system test cases are reviewed and revised.
<b>Outputs</b>	The output from this task is the SSTP with reviewed system level test cases.
<b>References</b>	None.

**Task 4: Hold Design Brainstorming Meetings**

<b>Objectives</b>	The purpose of this task is to generate a list of design issues and determine a set of design alternatives for each issue. The brainstorming meetings can also function as a tutorial for consultants from other development organizations.
<b>Dependencies</b>	This task is dependent on Task 1.
<b>Responsibilities</b>	The design team leader is responsible for organizing the brainstorming meetings and giving a tutorial of the software product if requested by consultants.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• SDP</li><li>• SRS</li><li>• SHA</li><li>• SOW</li></ul> <p>In addition, the following input may be included if appropriate:</p> <ul style="list-style-type: none"><li>• System Specifications</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>The software project manager organizes a series of brainstorming meetings in which the design team, along with the appropriate consultants, discuss the high-level design issues which need to be addressed. The first of these meetings is devoted to exploring exactly what decisions need to be made, what decisions have already been dictated, and any interdependencies among the various design issues. The inputs to this task are possible sources of decisions previously made. Design issues might include the following:</p> <ul style="list-style-type: none"><li>• What hardware platform will be used for the product?</li><li>• What software development tools will be used to produce the product?</li><li>• Will the product make use of any commercial or otherwise pre-existing software?</li><li>• What design methodology will be employed in the development of the product (i.e. structured, object-oriented, etc.)?</li><li>• What philosophy will be used to ensure product surety (i.e. functional compartmentalization, temporal compartmentalization, redundancy, etc.)?</li><li>• Is there a consistent error-handling philosophy?</li><li>• How will the design be documented?</li><li>• When will the design be considered sufficiently specified to begin implementation?</li><li>• How will the design be affected by the system specifications?</li></ul> <p>Once a list of issues has been generated, the software project manager schedules as many meetings as necessary to discuss and brainstorm possible solutions for each issue. Each issue may be considered separately or in conjunction with other issues if significant interdependencies exist.</p> <p>The software project manager appoints a recorder and a moderator for each meeting. The recorder is responsible for producing the minutes of the meeting. The moderator is responsible for keeping the meeting on track and seeing that all participants have an opportunity to express their ideas. Strengths and weaknesses of a given idea may be discussed in order to fuel the brainstorming process, but the moderator should steer the</p>

## Design Phase

meeting so that it is clear that the objective is to generate ideas, not make decisions.

One design issue which must be addressed is: How will the product functions be partitioned at the highest level? The choices of design methodology and surety philosophy impact this decision. Alternatives are explored and documented in the same manner as for other design issues.

### **Exit Criteria**

The design team agrees that all design issues have been identified and explored.

### **Outputs**

The outputs from this task are:

- the list of design issues,
- the list of alternatives for each design issue, and
- the minutes of brainstorming meetings

### **References**

None.

**Task 5: Investigate and Evaluate Alternative Designs**

<b>Objectives</b>	The purpose of this task is to analyze the various alternatives for each design issue and determine the best design for the software product. In addition, a high-level design of the software is produced.
<b>Dependencies</b>	This task is dependent on Task 4.
<b>Responsibilities</b>	The design team is responsible for analyzing and evaluating design alternatives.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• List of design issues, produced in Task 4</li><li>• List of alternatives for each design issue, produced in Task 4</li><li>• Minutes of brainstorming meetings, produced in Task 4</li><li>• SHA</li><li>• SOW</li><li>• SRS</li></ul> <p>In addition, the following input may be included if appropriate:</p> <ul style="list-style-type: none"><li>• System Specifications</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>Completion of this task requires that the design team produce a decision paper for each design issue identified in Task 4. These decision papers are included in the SDD. A decision paper includes the following, as appropriate:</p> <ul style="list-style-type: none"><li>• assumptions</li><li>• interfaces and dependencies</li><li>• description of each design alternative</li><li>• list of strengths and weaknesses of each alternative</li><li>• design hazard analysis on each alternative, performed by the surety expert</li><li>• impact of each design alternative on testing</li><li>• risk analysis on each alternative</li><li>• cost/benefit analysis of each alternative</li><li>• decision on which alternative is considered the best</li></ul> <p>In some cases, simulations or prototypes may be utilized to explore design alternatives and generate data for the decision papers. Outside expertise may be sought as well and consultants from applicable areas may be utilized in the decision making process.</p> <p>Any software development tools and commercial or otherwise pre-existing software that will be utilized are itemized. Procurement of tools and commercial software is initiated.</p> <p>An approach for the high-level partition of the product functions is also chosen at this time. Based on this approach the product functions, as specified in the SRS and other specifications, are partitioned to form a high-level design of the software.</p>
<b>Exit Criteria</b>	All design decision papers are completed and approved by the design team, and a high-level design of the software is produced.

## Design Phase

### **Outputs**

The outputs from this task are:

- design decision papers,
- development tools,
- existing code, and
- high-level software design.

### **References**

None.

## Design Phase

### **Task 6: Produce Design**

<b>Objectives</b>	The purpose of this task is to add detail to the high-level software design.
<b>Dependencies</b>	This task is dependent on Task 5 and is performed in parallel with Task 7.
<b>Responsibilities</b>	The design team is responsible for producing the design.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• Design decision papers, produced in Task 5</li><li>• High-level software design, produced in Task 5</li><li>• RTM</li><li>• SRS</li><li>• SOW</li><li>• Surety Folder</li></ul> <p>In addition, the following input may be included if appropriate:</p> <ul style="list-style-type: none"><li>• System Specifications</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>Completion of this task requires that the design team generate some representation of the design (could be graphical) and a decomposition of the product based on the partition chosen in Task 5 including input and output specifications. The team also investigates system integrity and recovery interfaces and considers the robustness, maintainability and modularity (cohesion/coupling) of the proposed design.</p> <p>The detailed design may be expressed in many ways including:</p> <ul style="list-style-type: none"><li>• data models</li><li>• process models</li><li>• flow charts</li><li>• structure charts</li><li>• data dictionary</li><li>• pseudocode</li><li>• state transition diagrams</li><li>• data structure diagrams</li></ul> <p>As the design team produces the decomposition of the design, the RTM is updated to include a mapping from the requirements to the design modules.</p>
<b>Exit Criteria</b>	The design is specified to a sufficient level of detail to allow for unambiguous implementation and the RTM has been updated.

## Design Phase

### **Outputs**

The outputs from this task are:

- the detailed design and
- the RTM with mapping of requirements to design modules.

### **References**

None.

**Task 7: Perform Design Hazard Analysis (DHA)**

<b>Objectives</b>	The purpose of this task is to perform a hazard analysis of the design.
<b>Dependencies</b>	This task is dependent on Task 5 and is performed in parallel with Task 6.
<b>Responsibilities</b>	The surety expert is responsible for performing the DHA.
<b>Inputs</b>	The inputs to this task are: <ul style="list-style-type: none"><li>• Detailed design information as it becomes available, from Task 6</li><li>• RTM with mappings to design modules as it becomes available, from Task 6</li><li>• Surety Folder</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>As the design team adds to the RTM, the surety expert adds to the Surety Folder a mapping from the surety requirements to design modules and identifies all design modules which have an effect on the surety functions of the product.</p> <p>As detailed design information becomes available, the surety expert performs a hazard analysis of the design. This includes analyzing surety-critical software components in order to assess their degree of risk and relationships to other components. Any findings are input back into Task 6 in order to resolve surety issues which will eliminate hazards or mitigate the risk of hazards with respect to the design.</p>
<b>Exit Criteria</b>	The DHA is complete and the Surety Folder is updated to include a mapping from surety requirements to design modules as well as a list of all surety critical modules.
<b>Outputs</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• the DHA and</li><li>• the Surety Folder with mapping of surety requirements to design modules and identification of surety critical modules.</li></ul>
<b>References</b>	<ul style="list-style-type: none"><li>• IEEE Std. 1228 [10]</li><li>• MIL-STD-882B [3]</li><li>• Software Systems Safety Handbook [26]</li><li>• A Study on Hazard Analysis in High Integrity Software Standards and Guidelines [27]</li><li>• NASA Software Safety Standard [29]</li></ul>

## Design Phase

### **Task 8: Complete SDD**

<b>Objectives</b>	The purpose of this task is to complete the SDD.
<b>Dependencies</b>	This task is dependent on Task 6.
<b>Responsibilities</b>	The design team is responsible for writing the SDD.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• SDD Template</li><li>• Decision Papers produced in Task 5</li><li>• Detailed Design generated in Task 6</li><li>• RTM</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Completion of this task requires that all applicable sections of the SDD are written. The outputs from Task 5 and Task 6 are incorporated in the SDD where appropriate. Some information may be placed in appendices.
<b>Exit Criteria</b>	The SDD is complete and is placed under configuration management according to the SCMP.
<b>Outputs</b>	The output from this task is the SDD.
<b>References</b>	IEEE MIL-Std. 498 [2]

## Design Phase

### **Task 9: Inspect SDD**

The Inspection Process is described in detail in Section 5.1 of this document. The following paragraphs describe the particulars of inspecting the SDD.

<b>Inspection Materials</b>	SDD
<b>Task Dependencies</b>	This task is dependent on Task 8.
<b>Support Materials</b>	The following materials may be supplied to the inspection team as supporting material for the inspection: <ul style="list-style-type: none"><li>• DHA</li><li>• SOW</li><li>• SRS</li><li>• Surety Folder</li><li>• System Specifications</li></ul>
<b>Inspection Team</b>	The inspection team should include the surety expert, a test team representative, a requirements analyst, and some members of the design team.
<b>General Objectives</b>	All inspection team members are responsible for verifying that the requirements have been interpreted consistently and correctly and that all requirements are met by the design. The SDD itself is examined for consistency, completeness, and comprehensiveness and that it conforms to the SDD template identified in the Project Planning Phase.
<b>Testing Objectives</b>	The test team representative is primarily responsible for ensuring that the design is testable.
<b>Surety Objectives</b>	The surety expert is primarily responsible for ensuring that the surety objectives of the inspection are met. The surety objectives for the inspection include the verification of the surety portion of the RTM and the surety critical modules. It is also confirmed that the all hazards identified in the DHA have been addressed.
<b>Output</b>	The output from this task is the revised SDD.

## Design Phase

### **Task 10: Produce Integration Test Plan**

<b>Objectives</b>	Produce integration test plan for execution during the Integration Phase.
<b>Dependencies</b>	This task is dependent on Task 5 and utilizes the detailed design from Task 6, as it becomes available.
<b>Responsibilities</b>	The MTP defines who is responsible for integration testing.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• Detailed design information as it becomes available, from Task 6</li><li>• DHA</li><li>• High-level software design, produced in Task 5</li><li>• MTP</li><li>• SRS</li><li>• SSTP</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start. The team responsible for developing the integration test plan must be assembled.
<b>Task Description</b>	<p>First, an integration approach is selected. The high-level design is then divided into integration units according to the selected approach. The most common approaches to integration are:</p> <ul style="list-style-type: none"><li>• big-bang - combine all units at once and test the complete system as an entity without considering the individual components.</li><li>• top-down - start at the top of the high-level design and add units working down to those at the bottom. Units that are not yet integrated are replaced by stubs. This approach may be the most effective way to approach integration testing.</li><li>• bottom-up - start at the bottom of the high-level design and add units working up to the top. This approach addresses the most significant components early in the test process which allows for redesign of component interfaces early but flaws in the design may show up much later. Units that are not yet integrated are replaced by drivers.</li><li>• phased - sometimes called thread integration, this approach uses the SRS or system specification to identify complete functions or threads (input-process-output definitions) and then test the integration of all threads.</li></ul> <p>Next, the team identifies and defines the interfaces within each integration unit and between integration units in terms of expected inputs and outputs. If any system-level requirements are contained within a given integration unit, test cases from the SSTP may be utilized in the development of integration tests. Once expected inputs and outputs are defined, pass/fail criteria and test methods are determined. Finally, test procedures are written.</p> <p>The integration test plan includes the integration approach and decomposition, expected inputs and outputs, pass/fail criteria, test methods, and the test procedures. The integration test plan may be documented either as a section of the SSTP or in a software development folder. The MTP should specify where the documentation will reside.</p>

## Design Phase

<b>Exit Criteria</b>	Integration test plan is complete.
<b>Outputs</b>	The output from this task is the integration test plan.
<b>References</b>	None.

**Task 11: Review/Inspect Integration Test Plan**

<b>Objectives</b>	The purpose of this task is to review and/or inspect the integration test plan.
<b>Dependencies</b>	This task is dependent on Task 10.
<b>Responsibilities</b>	The MTP defines who is responsible for integration testing. The surety expert reviews the plan for adequate coverage of surety-critical areas.
<b>Inputs</b>	<p>The inputs to this task are:</p> <ul style="list-style-type: none"><li>• High-level software design produced in Task 5</li><li>• Integration test plan</li><li>• MTP</li><li>• Surety Folder</li></ul> <p>In addition, the following inputs may be included as references:</p> <ul style="list-style-type: none"><li>• SDD</li><li>• SSTP</li><li>• Detailed design produced in Task 6.</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	The MTP specifies whether the integration test plan will be inspected, or if a less formal review is sufficient. In either case, those responsible for integration testing review the integration test plan for completeness and accuracy. The surety expert ensures that the plan gives adequate coverage of the surety requirements.
<b>Exit Criteria</b>	The integration test plan is reviewed and revised.
<b>Outputs</b>	The output from this task is the integration test plan.
<b>References</b>	None.

### 3.4 Implementation Phase

#### 3.4.1 Overview

The Implementation Phase uses the blueprint provided by the Design Phase to produce code units which have been unit-tested. The Implementation Phase provides the code units to the Integration Phase where the units are combined and tested. The Implementation Phase provides input primarily in the form of source code to the following phases:

- Integration
- Test

#### 3.4.2 Inputs and Suppliers

The inputs to the Implementation Phase are summarized in Table 3-11.

**Table 3-11 - Implementation Phase Inputs and Suppliers**

<b>Input</b>	<b>Supplier</b>
Development Tools	Design Phase
DHA	Design Phase
Existing Code (libraries, COTS)	Design Phase
Existing Coding Standards (if available)	Project Planning Phase (SDP)
Modification Requests	Various Sources
MTP	Project Planning Phase
RTM	Design Phase
SDD	Design Phase
SDP	Project Planning Phase
SOW	Customer
SRS	Requirements Phase
Surety Folder	Design Phase
System Specifications	System Engineers

#### 3.4.3 Outputs and Customers

Table 3-12 contains the outputs from the Implementation Phase.

## Implementation Phase

**Table 3-12 - Implementation Phase Outputs and Customers**

<b>Output</b>	<b>Customer</b>
Metrics	Quality Assurance, Project Manager, Other Projects
Modification Requests	Various Sources
RTM	Integration Phase
SDF	Integration Phase
Surety Folder	Integration Phase
Source Code	Integration Phase
Unit Tests	Integration Phase

### 3.4.4 Metrics

This phase's metrics should provide a measure of the effectiveness of the Implementation Phase and include:

- code inspection metrics as reported on the Inspection Summary form and
- reliability metrics from formal unit testing.

### 3.4.5 Task Summary

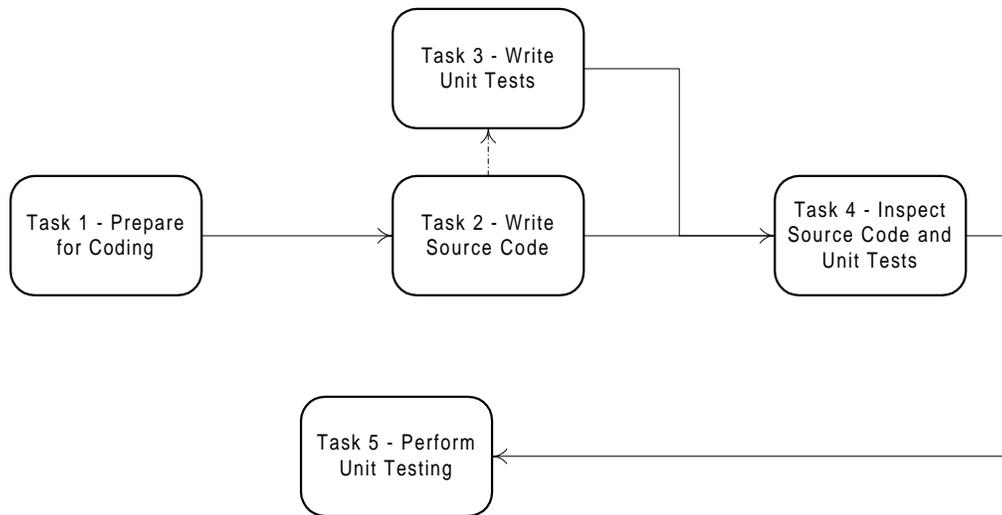
The Implementation Phase tasks are summarized in Table 3-13. Each of these tasks is described in greater detail in Section 3.4.8.

**Table 3-13 - Implementation Phase Task Summary**

<b>Task</b>	<b>Description</b>	<b>Dependencies</b>
1	Prepare For Coding	
2	Write Source Code	1
3	Write Unit Tests	2
4	Inspect Source Code and Unit Tests	2,3
5	Perform Unit Testing	4

## Implementation Phase

### 3.4.6 Task Flow



**Figure 3-4: Implementation Phase Task Flow**

### **3.4.7 Modification Requests**

#### **3.4.7.1 Generation of Requests**

There are several tasks performed during the Implementation Phase in which modification requests may be generated which affect the outputs of other phases. For instance, modification requests may be submitted to the Design Phase in order to modify the SDD. This may become necessary if at any time during source code writing in Task 2, a design feature is be found to be misstated or not codeable. Modification requests may also be submitted to the Project Planning Phase if changes are found to be necessary to the schedule or resource allocations specified in the SDP.

#### **3.4.7.2 Receipt and Processing of Requests**

The Implementation Phase may receive modification requests in order to update an output from the Implementation Phase. These modification requests are handled by revisiting appropriate tasks from the Implementation Phase. When tasks are revisited, the time devoted to each task will depend on the scope of the modification request. Tasks which may have taken weeks to perform during the initial completion of the Implementation Phase may take only hours to perform when addressing a modification request. If a modification request is received during the initial performance of the Implementation Phase, only those tasks which have been completed need to be revisited. Tasks in progress or not yet begun are performed as usual taking the new information from the modification request into account.

### 3.4.8 Detailed Task Descriptions

This section provides the detailed description of each task listed in Table 3-13. Each task should be completed to successfully carry out the Implementation Phase of development. Tasks 2 through 5 can begin for a unit of code regardless of the completion state of other units of code.

#### **Task 1: Prepare for Coding**

<b>Objectives</b>	The purpose of this task is to ensure the software developers are adequately prepared to begin developing code.
<b>Dependencies</b>	None.
<b>Responsibilities</b>	The software developers are responsible for all activities in this task.
<b>Inputs</b>	<p>The inputs for this task are:</p> <ul style="list-style-type: none"> <li>• Coding Standards (specified in the SDP)</li> <li>• DHA</li> <li>• SDD</li> <li>• SDP</li> <li>• SRS</li> <li>• Surety Folder</li> </ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"> <li>• Development Tools</li> <li>• Existing Code</li> <li>• System Specifications</li> </ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	The design modules specified in the SDD are allocated among the software developers for implementation. The developer becomes familiar with the input items. The Surety Folder and the DHA from the Design Phase provides a list of hazards (surety concerns), surety requirements, specific output states to avoid, and common coding defects applicable to the specific code to be written. The coding standards may be revised to fit the project and may incorporate findings from the surety folder including surety concerns (things to avoid) and common programming errors. The defects and limitations of the development tools are researched and the findings placed into the Surety Folder. The development tools are gathered to form a properly configured Software Engineering Environment (SEE).
<b>Exit Criteria</b>	The coding assignments have been made and the SEE is configured.
<b>Outputs</b>	<p>The outputs from this task are:</p> <ul style="list-style-type: none"> <li>• coding standards,</li> <li>• the Surety Folder, and</li> <li>• the SEE.</li> </ul>
<b>References</b>	None.

## Implementation Phase

### **Task 2: Write Source Code**

<b>Objectives</b>	The purpose of this task is to produce source code that accurately represents the SDD, and that satisfies or otherwise supports all applicable requirements of the SRS.
<b>Dependencies</b>	This task is dependent on Task 1.
<b>Responsibilities</b>	The software developers are responsible for code development. The surety expert is responsible for a code-level hazard analysis.
<b>Inputs</b>	<p>The inputs for this task are:</p> <ul style="list-style-type: none"><li>• Coding Standards</li><li>• DHA</li><li>• SDD</li><li>• SDP</li><li>• SEE</li><li>• Surety Folder</li></ul> <p>In addition, the following inputs may be included if appropriate:</p> <ul style="list-style-type: none"><li>• Existing Code</li><li>• SOW</li><li>• System Specifications</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>Execution of this task requires completion of the following steps:</p> <ol style="list-style-type: none"><li>1. Develop commented unit source code that implements the design in the SDD and that addresses surety concerns in the surety folder and coding standards.</li><li>2. Compile code and correct syntax defects. It is suggested the developers use the highest warning levels possible for compilers.</li><li>3. Using other available tools (e.g., lint) check for and correct other defects or warnings (if applicable).</li><li>4. If reliability metrics are to be used to assess the risk associated with this code, the developer should collect defects data (# of defects, execution time-to-failure, etc.) during code development. These data are placed into the SDF and may be used to fit a software reliability model and quantify the reliability of this code.</li><li>5. During code development, a code-level hazard analysis is performed by the surety expert. This analysis covers the actual source code, object code, system interfaces, and software documentation (to ensure surety requirements are included). Findings that impact surety should be included in an updated surety folder. Based on these findings, the software design, code, and test cases are modified to eliminate or mitigate the risk of the discovered hazards.</li><li>6. Update the RTM to trace source modules to their design modules.</li></ol>
<b>Exit Criteria</b>	The software developers ensure that the code compiles cleanly and all hazards identified by the surety expert, if any, are addressed. The code is then placed under configuration management according to the SCMP.

## Implementation Phase

### Outputs

The outputs from this task are:

- the RTM,
- the SDF,
- the Surety Folder, and
- Source Code.

### References

- MIL-STD-882B [3]
- IEEE 1228 [10]
- Software Systems Safety Handbook [26]
- A Study on Hazard Analysis in High Integrity Software Standards and Guidelines [27]
- NASA Software Safety Standard [29]

## Implementation Phase

### **Task 3: Write Unit Tests**

<b>Objectives</b>	The purpose of this task is to write the unit tests.
<b>Dependencies</b>	This task is performed in parallel with Task 2.
<b>Responsibilities</b>	The software developer and/or test engineer are responsible for unit test development as defined in the MTP.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• SDD</li><li>• Source Code</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Using information gained from code development in Task 2, the detailed test procedures for the formal unit testing should be written. This may involve, for example, the design and production of test data, stubs, drivers, and the development of procedures for using emulation equipment.
<b>Exit Criteria</b>	The unit test procedures are placed under configuration management according to the SCMP.
<b>Outputs</b>	The output from this task is the detailed test procedures for unit testing.
<b>References</b>	IEEE Std 1008-1987 [7]

#### **Task 4: Inspect Source Code and Unit Tests**

The Inspection Process is described in detail in Section 5.1 of this document. The following paragraphs describe the particulars of inspecting the code and unit test procedures.

<b>Inspection Materials</b>	Source code and associated unit test procedures
<b>Task Dependencies</b>	This task is dependent on Task 2 and Task 3.
<b>Support Materials</b>	<p>The following materials may be supplied to the inspection team as supporting material for the inspection:</p> <ul style="list-style-type: none"><li>• Coding Standards (as documented in the SDP)</li><li>• SDD</li><li>• SRS</li><li>• Surety Folder</li></ul>
<b>Inspection Team</b>	The inspection team should include the surety expert, a test engineer, design team members, as well as the authors of the source code and unit tests, and other software developers.
<b>General Objectives</b>	The inspection team members are responsible for verifying that the code represents the SDD and satisfies all applicable requirements in the SRS.
<b>Testing Objectives</b>	The test engineer is primarily responsible for becoming familiar with the code and determining if the unit tests are sufficient. The software developers are responsible for verifying that the unit tests are feasible.
<b>Surety Objectives</b>	The surety expert reviews the code during the inspection to assure that the actual code correctly implements the intent of the surety requirements and that no additional hazards are introduced. The surety expert updates the surety folder to identify specific surety-critical code features that should be tested and creates a surety defects report from the inspection.
<b>Outputs</b>	<p>The outputs of the inspection are:</p> <ul style="list-style-type: none"><li>• the inspection reports,</li><li>• the revised code and associated formal unit test procedures,</li><li>• the updated surety folder, and</li><li>• a surety defects report.</li></ul>

## Implementation Phase

### **Task 5: Perform Unit Testing**

<b>Objectives</b>	The purpose of this task is to test the implementation and design details of a code unit.
<b>Dependencies</b>	This task is dependent on Task 4.
<b>Responsibilities</b>	The software developer and/or test engineer are responsible for testing the code at the unit level as defined in the MTP. The software developers are responsible for making necessary corrections. The surety expert is responsible for ensuring that tests of surety-critical software components are conducted in strict accordance with the software unit test plans.
<b>Inputs</b>	The input for this task is the code to be tested and the corresponding detailed unit test procedures.
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>This task is conducted according to the appropriate unit test procedures. If time is limited, priority should be given to testing the surety aspects of the code. This includes the thorough testing of surety-critical software units under normal conditions and abnormal environment and input conditions. Documentation of the testing includes the following:</p> <ul style="list-style-type: none"><li>• time to failure (for showing reliability growth)</li><li>• failure classification</li><li>• history of failure (what test case caused failure)</li><li>• conditions/environment of failure</li><li>• reasons why failure wasn't detected earlier</li><li>• resolution and disposition of failure (includes regression testing)</li></ul> <p>This documentation should be placed into the Software Development Folder (SDF).</p>
<b>Exit Criteria</b>	The exit criteria is resolution of all detected failures.
<b>Outputs</b>	The output is fully unit-tested source code.
<b>References</b>	<ul style="list-style-type: none"><li>• MIL-STD-882B [3]</li><li>• IEEE Std 1008-1987 [7]</li><li>• Software Systems Safety Handbook [26]</li><li>• A Study on Hazard Analysis in High Integrity Software Standards and Guidelines [27]</li><li>• NASA Software Safety Standard [29]</li></ul>

### 3.5 Integration Phase

#### 3.5.1 Overview

The purpose of the Integration Phase is to determine how individual software units perform together as an integrated system. Depending on the size of the system, there may be several levels of integration tests. For example, units are first integrated into components, components are integrated into subsystems, and then subsystems are combined to form the total software system.

It is not the purpose of the integration tests to revalidate requirements previously verified during lower-level tests but rather to build on previous test and evaluation. Integration testing should focus on the interfaces between software units. If hardware items are available, integration testing should also include testing the interfaces between software units and hardware.

For HCSs, this phase should ensure that as the software units are integrated, a hazardous state, as identified in the DHA, does not occur. The Integration Phase provides input primarily in the form of an SSTP to the Test Phase.

#### 3.5.2 Inputs and Suppliers

The inputs to the Integration Phase are summarized in Table 3-14.

**Table 3-14 - Integration Phase Inputs and Suppliers**

<b>Input</b>	<b>Supplier</b>
DHA	Design Phase
Integration Test Plan	Design Phase
Modification Requests	Various Sources
MTP	Project Planning Phase
RTM	Implementation Phase
SDD	Design Phase
SDF	Implementation Phase
SDP	Project Planning Phase
SRS	Requirements Phase
SSTP	Design Phase
Surety Folder	Implementation Phase
Source Code	Implementation Phase

## Integration Phase

### 3.5.3 Outputs and Customers

Table 3-15 contains the outputs from this process.

**Table 3-15 - Integration Phase Outputs and Customers**

<b>Output</b>	<b>Customer</b>
Metrics	Project Manager, Quality Assurance, Other Projects
Modification Requests	Various Phases
SDF	Test Phase
Source Code	Test Phase
SSTP	Test Phase
Surety Folder	Test Phase
RTM	Software Project Manager

### 3.5.4 Metrics

This phase's metrics should provide a measure of the effectiveness of the Integration Phase and include:

- reliability metrics and
- test passage metrics such as percent first pass, percent second pass, etc., and percent failure after passage.

### 3.5.5 Task Summary

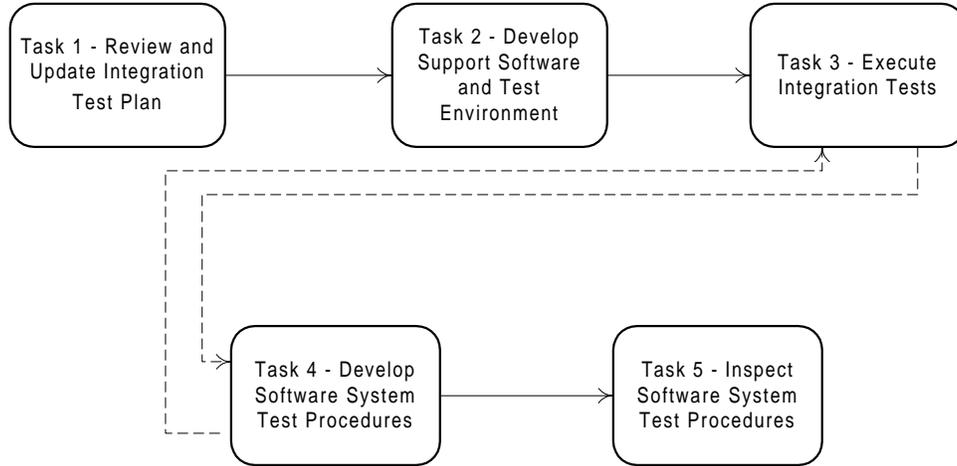
The Integration Phase tasks are summarized in Table 3-16. Each of these tasks is described in greater detail in Section 3.5.7.

**Table 3-16 - Integration Phase Task Summary**

<b>Task</b>	<b>Description</b>	<b>Dependencies</b>
1	Review and Update Integration Test Plan	
2	Develop Support Software and Test Environment	1
3	Execute Integration Tests	2
4	Develop Software System Test Procedures	
5	Inspect Software System Test Procedures	4

## Integration Phase

### 3.5.6 Task Flow



**Figure 3-5 - Integration Phase Task Flow**

### **3.5.7 Modification Requests**

#### **3.5.7.1 Generation of Requests**

During the Integration Phase, modification requests are generated which can affect the outputs of other phases. For instance, if a requirement is found to be incorrect during integration testing, a modification request may be submitted to the Requirements Phase in order to modify the SRS. In addition, modification requests may also be submitted to the Design Phase to change the SDD and to the Implementation Phase to change the code and unit test plan.

#### **3.5.7.2 Receipt and Processing of Requests**

The Integration Phase receives modification requests to: 1) update the integration test plan and re-execute integration testing, or 2) update the software system test procedures. These modification requests are handled by revisiting the appropriate tasks from the Integration Phase. When tasks are revisited, the time devoted to each task will depend on the scope of the modification request. Tasks which may have taken weeks to perform during the initial completion of the Integration Phase may take only hours to perform when addressing a modification request. If a modification request is received during the initial completion of the Integration Phase, only those tasks which have been completed need to be revisited. Tasks in progress or not yet begun are performed as usual taking the new information from the modification request into account.

### 3.5.8 Detailed Task Descriptions

This section provides the detailed description of each task listed in Table 3-16. Each task should be completed to successfully carry out the Integration Phase of development.

#### **Task 1: Review and Update Integration Test Plan**

<b>Objectives</b>	The purpose of this task is to review and update the integration test plan created during the Design Phase.
<b>Dependencies</b>	None.
<b>Responsibilities</b>	Depending upon how integration testing is defined in the MTP, either a software developer or a test engineer is responsible for all activities defined by this task.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• Integration test plan</li><li>• MTP</li><li>• SDD</li><li>• SDFs with results of unit testing</li><li>• Surety Folder with DHA</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	The software developer or test engineer <ul style="list-style-type: none"><li>• reviews the integration test plan created during the Design Phase against the MTP, the SDD, and unit test results to ensure completeness and correctness.</li><li>• verifies surety concerns identified in the DHA have been addressed by the integration test plan.</li><li>• updates the integration test plan.</li></ul>
<b>Exit Criteria</b>	The integration test plan is reviewed and, if necessary, updated.
<b>Outputs</b>	The output from this task is an updated integration test plan.
<b>References</b>	None.

**Task 2: Develop Support Software and Test Environment**

<b>Objectives</b>	The purpose of this task is to develop any support software, e.g., test stubs, test drivers, that are needed to support integration testing and to prepare the test environment, e.g., all necessary hardware, software, tools, data sets, and personnel necessary to execute the integration tests.
<b>Dependencies</b>	This task is dependent on Task 1.
<b>Responsibilities</b>	Depending upon how integration testing is defined in the MTP, either a software developer or a test engineer is responsible for all activities defined by this task.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• Integration test plan</li><li>• Source Code under configuration management</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>The software developer or test engineer, using the integration test plan, develops support software to allow the testing of software units as they are integrated together to form larger components.</p> <p>If a top-down approach is taken to integration testing, then test stubs will need to be developed for those units below the units being tested.</p> <p>If a bottom-up approach is taken, then test drivers will need to be developed to act as the calling routines for lower level units.</p> <p>If phased integration or big bang is used, support software may not be required. The software developer or test engineer combines support software with the source code to produce integration test code.</p> <p>In preparing the test environment, the software developer or test engineer should:</p> <ul style="list-style-type: none"><li>• Determine the tests to be run</li><li>• Determine the hardware, tools, and data sets required</li><li>• Schedule the necessary testing personnel</li><li>• Gather the hardware, tools, and data sets required</li><li>• Assemble the integration test plan, Test Log, Incident Report, and any other documentation required</li><li>• Install the executable code to be tested</li></ul>
<b>Exit Criteria</b>	Support software is developed and the test environment is established.
<b>Outputs</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• required support software and</li><li>• the test environment.</li></ul>
<b>References</b>	None.

## Integration Phase

### **Task 3: Execute Integration Tests**

<b>Objectives</b>	<p>The purpose of this task is to:</p> <ul style="list-style-type: none"><li>• verify software/software interactions and</li><li>• ensure that the software meets customer requirements.</li></ul>
<b>Dependencies</b>	<p>This task is dependent on Task 2 and interdependent on Task 4.</p>
<b>Responsibilities</b>	<p>Depending upon how integration testing is defined in the MTP, either a software developer or a test engineer is responsible for all testing activities defined by this task. The software developers are responsible for making necessary corrections to the source code. The surety expert is responsible for ensuring that tests of surety-critical software components are conducted in strict accordance with the software test plans and procedures.</p>
<b>Inputs</b>	<p>The inputs for this task are:</p> <ul style="list-style-type: none"><li>• Integration test plan</li><li>• MTP</li><li>• SDF or SSTP</li><li>• support software</li><li>• Source Code to be integrated and tested</li></ul>
<b>Entrance Criteria</b>	<p>The inputs must be available before this task can start. The test environment and necessary tools must be available.</p>
<b>Task Description</b>	<p>The software developer or test engineer:</p> <ol style="list-style-type: none"><li>1. Prior to each test session, determines the tests to be run, and the tools and data sets required.</li><li>2. During each test session, runs tests according to the plan for integration of software units and documents test results in the SDF or in the SSTP. Note: Test results with an impact on surety are also documented in the surety folder.</li><li>3. After each test session, analyzes test results and completes incident reports as necessary. Note: Incident reports with an impact on surety are documented in the surety folder along with their resolution.</li><li>4. After each test session, if errors were found, the software developer corrects the code.</li></ol> <p>The four activities above are repeated until testing is completed without incidents.</p> <p>The surety expert will ensure that deficiencies and discrepancies are corrected and retested.</p> <p>If reliability metrics are to be used to assess the risk associated with this code, the software developer or test engineer collects reliability metrics during integration testing. These metrics are placed into the SDF and may be used to fit a software reliability model and quantify the reliability of this code.</p> <p>This task is conducted according to the integration test plan. If time is limited, priority is given to testing the surety aspects of the code. Documentation of the testing includes the following:</p> <ul style="list-style-type: none"><li>• time to failure (for showing reliability growth)</li></ul>

## Integration Phase

- failure classification
- history of failure (what test case caused failure)
- conditions/environment of failure
- reasons why failure wasn't detected earlier
- resolution and disposition of failure (includes regression testing)

This documentation is placed into the SDF or the SSTP.

*Note: The MTP defines whether the integration test results and incident reports are documented in the SDF or the SSTP.*

### **Exit Criteria**

All integration tests defined by the integration test plan have executed successfully or all incidents have been resolved before exiting this task.

### **Output**

The output from this task is the SDF and/or the SSTP with documentation of all integration testing and reliability metrics. In addition, updated source code is also an output from this task.

### **References**

- MIL-STD-882B [3]
- IEEE 1228 [10]
- A Study on Hazard Analysis in High Integrity Software Standards and Guidelines [27]
- NASA Software Safety Standard [29]

**Task 4: Develop Software System Test Procedures**

<b>Objectives</b>	The purpose of this task is to produce the software system test procedures and document them in the SSTP.
<b>Dependencies</b>	This task is interdependent on Task 3.
<b>Responsibilities</b>	The test engineer is responsible for all activities defined by this task.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• RTM</li><li>• SDD</li><li>• SRS</li><li>• SSTP (includes test cases)</li><li>• Surety Folder</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<p>This task consists of defining test procedures for all software system test cases defined in the SSTP.</p> <p>Test procedures should include a description of:</p> <ul style="list-style-type: none"><li>• the testing approach (e.g. intrusive, non-intrusive)</li><li>• the test sequence,</li><li>• the test input data,</li><li>• the software configuration,</li><li>• the test equipment, and</li><li>• the required test personnel and their function.</li></ul> <p>The test procedures are documented in the SSTP.</p> <p>Surety requirements are mapped to specific surety tests. Testing surety requirements often requires intrusive techniques and cannot be tested in a black box mode. A surety testing matrix is created which maps surety requirements to a testing approach, e.g., intrusive, non-intrusive.</p>
<b>Exit Criteria</b>	The test engineer verifies that all test cases are covered by test procedures.
<b>Outputs</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• SSTP updated with software system test procedures,</li><li>• RTM (with test procedures traced to test cases), and</li><li>• Surety Folder with updated surety requirements trace matrix (with test procedures traced to test cases).</li></ul>
<b>References</b>	None.

**Task 5: Inspect Software System Test Procedures**

<b>Inspection Materials</b>	SSTP
<b>Task Dependencies</b>	This task is dependent on Task 4.
<b>Support Materials</b>	<p>The following materials should be supplied to the inspection team as supporting material for the inspection:</p> <ul style="list-style-type: none"><li>• RTM</li><li>• SDD</li><li>• SRS</li><li>• Surety Folder</li></ul>
<b>Inspection Team</b>	The inspection team should include the surety expert, the software test manager, the software project manager, and a member from each development team (requirements, design, implementation, integration, and test).
<b>General Objectives</b>	<p>The review/inspection team is responsible for verifying that the test procedures:</p> <ul style="list-style-type: none"><li>• conform to the SSTP template selected during the Project Planning Phase</li></ul>
<b>Testing Objectives</b>	<p>The review/inspection team is responsible for verifying the following:</p> <ul style="list-style-type: none"><li>• Test procedures cover 100% of the software system test cases</li><li>• Test procedures are described in enough detail to be reproducible and verifiable</li></ul>
<b>Surety Objectives</b>	The surety expert ensures that the software system test procedures adequately cover the surety requirements. Results from previous hazard analysis documented in the surety folder are used by the surety expert to perform the inspection.
<b>Outputs</b>	<p>The outputs from this task are:</p> <ul style="list-style-type: none"><li>• the inspection reports and</li><li>• the SSTP revised to include software system test procedures.</li></ul>

### 3.6 Test Phase

#### 3.6.1 Overview

The purpose of the Test Phase is to provide complete test coverage of all the software requirements stated in the SRS. Special emphasis must be placed on surety requirements. The Test Phase may also test hardware/software interactions and software/software interactions.

The Test Phase provides an SSTP to the customer and verification personnel as evidence that functional and surety requirements have been tested. The SSTP also identifies a set of regression tests that can be run on the software to ensure that modifications to the software have not inadvertently caused a defect.

Test Planning must occur as part of the Project Planning Phase. The result is an MTP that coordinates the testing activities that occur in all phases of software development. The MTP also addresses surety requirements and any special surety testing needs. See the Project Planning Phase for details.

Unit and Integration Testing are part of the software development process. See the Design, Implementation and Integration Phases for details on unit and integration testing.

#### 3.6.2 Inputs and Suppliers

The inputs to the Test Phase are summarized in the following table.

**Table 3-17 - Test Phase Inputs and Suppliers**

<b>Input</b>	<b>Supplier</b>
Modification Requests	Various Sources
MTP	Project Planning Phase
RTM	Integration Phase
SDD	Design Phase
SDF	Integration Phase
SDP	Project Planning Phase
SRS	Requirements Phase
SSTP	Integration Phase
Source Code	Integration Phase
Surety Folder	Integration Phase

### 3.6.3 Outputs and Customers

Table 3-18 contains the outputs from this process.

**Table 3-18 - Test Phase Outputs and Customers**

<b>Output</b>	<b>Customer</b>
Metrics	Quality Assurance, Project Manager, Other Projects
Modification Requests	Various Phases
SDF	Customer, Quality Assurance, Verification Agencies
SSTP	Customer, Quality Assurance, Verification Agencies
Source Code	Customer, Quality Assurance, Verification Agencies
Surety Folder	Customer, Project Manager, Quality Assurance, Verification Agencies

### 3.6.4 Metrics

This phase’s metrics should provide a measure of the effectiveness of the Test Process and include:

- SSTP inspection metrics as reported on the Inspection Summary form,
- reliability metrics, and
- test passage metrics such as percent first pass, percent second pass, etc., and percent failure after passage.

### 3.6.5 Task Summary

The Test Phase tasks are summarized in Table 3-19. Each of these tasks is described in greater detail in Section 3.6.7.

**Table 3-19 - Test Phase Task Summary**

<b>Task</b>	<b>Description</b>	<b>Dependencies</b>
1	Complete SSTP	
2	Inspect SSTP	1
3	Approve and Distribute SSTP	2
4	Produce Executable code and Prepare Test Environment	3
5	Execute Tests	4
6	Report and Document Incident Resolution	5
7	Rerun Tests	6
8	Update SSTP Results Section(s)	7

Test Phase

3.6.6 Task Flow

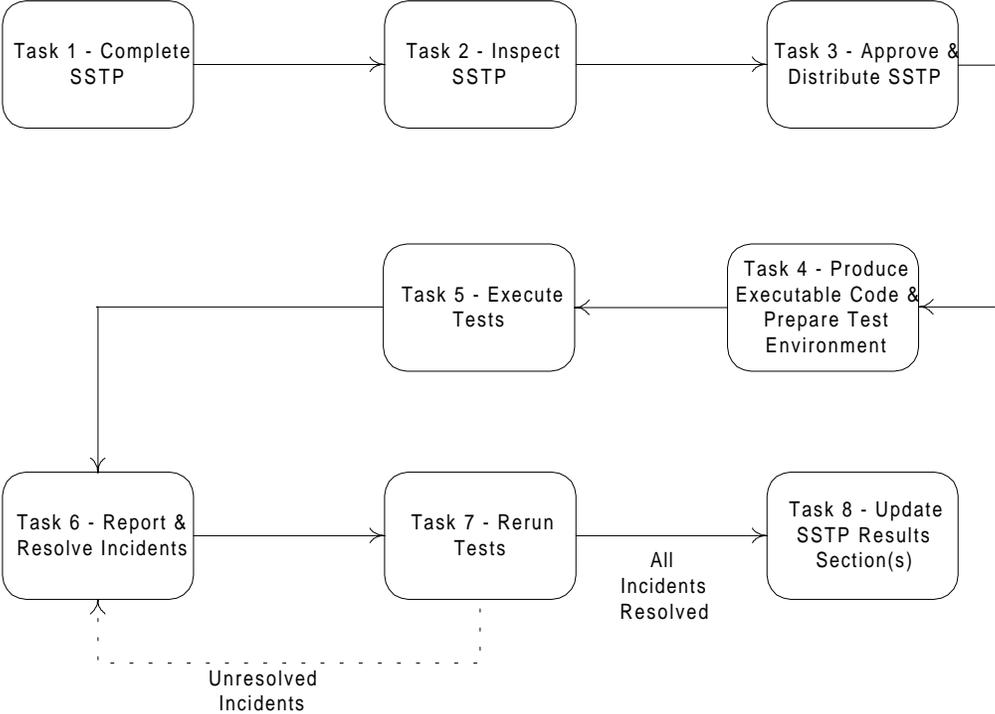


Figure 3-6: Test Phase Task Flow

### **3.6.7 Modification Requests**

#### **3.6.7.1 Generation of Requests**

There are several tasks performed during the Test Phase in which modification requests may be generated which affect the outputs of other phases. For instance, modification requests may be submitted to the Requirements Phase in order to modify the SRS. This may become necessary if, during the execution of software system test cases, a requirement is found to be incorrect. Modification requests may also be submitted to the Implementation Phase if code changes are necessary.

#### **3.6.7.2 Receipt and Processing of Requests**

The Test Phase may receive modification requests to update the SSTP. These modification requests are handled by revisiting appropriate tasks from the Test Phase. When tasks are revisited, the time devoted to each task will depend on the scope of the modification request. Tasks which may have taken weeks to perform during the initial completion of the Test Phase may take only hours to perform when addressing a modification request. If a modification request is received during the initial performance of the Test Phase, only those tasks which have been completed need to be revisited. Tasks in progress or not yet begun are performed as usual taking the new information from the modification request into account.

### 3.6.8 Detailed Task Descriptions

This section provides the detailed description of each task listed in Table 3-19. Each task should be completed to successfully carry out the Test Phase of development.

#### **Task 1: Complete SSTP**

<b>Objectives</b>	The purpose of this task is to complete the SSTP -- except for the test results.
<b>Dependencies</b>	None.
<b>Responsibilities</b>	The test engineer is responsible for all activities in this task.
<b>Inputs</b>	The input for this task is the SSTP.
<b>Entrance Criteria</b>	The inputs must be available and integration testing must be completed before this task can start.
<b>Task Description</b>	<p>Completion of this task requires that all applicable sections of the SSTP be written. Sections of the SSTP that require test results are not written until later in the Test Phase.</p> <p>A subset of software system tests is identified as regression tests.</p>
<b>Exit Criteria</b>	The moderator for the SSTP inspection (Task 2) verifies that the SSTP is ready for inspection. The SSTP must be placed under configuration management according to the SCMP.
<b>Outputs</b>	The output from this task is the SSTP (without test results).
<b>References</b>	None.

**Task 2: Inspect SSTP**

The Inspection Process is described in detail in Section 5.1 of this document. The following paragraphs describe the particulars of inspecting the SSTP.

<b>Inspection Materials</b>	SSTP
<b>Task Dependencies</b>	This task is dependent on Task 1.
<b>Support Materials</b>	The following materials are supplied to the inspection team as supporting material for the inspection: <ul style="list-style-type: none"><li>• SDD</li><li>• SRS</li><li>• Surety Folder</li></ul>
<b>Inspection Team</b>	The inspection team includes the surety expert, the software test manager, the software project manager, and a member from each development team (requirements, design, implementation, integration, and test).
<b>General Objectives</b>	The inspection team is responsible for verifying the following: <ul style="list-style-type: none"><li>• SSTP is complete and correct</li><li>• SSTP conforms to the SSTP template identified in the Project Planning Phase</li></ul>
<b>Testing Objectives</b>	The inspection team is responsible for verifying the following: <ul style="list-style-type: none"><li>• Tests provide 100 percent test coverage of the requirements in the SRS</li><li>• Test procedures are described in enough detail to be reproducible</li><li>• Expected results (pass/fail criteria) are described in enough detail to allow a conclusive pass/fail decision</li><li>• A subset of tests has been identified which can act as a regression suite for future testing activities</li></ul>
<b>Surety Objectives</b>	The inspection team is responsible for verifying the following: <ul style="list-style-type: none"><li>• Surety-related tests have been identified for all surety requirements</li><li>• Surety-related test cases have been identified and documented</li></ul>
<b>Outputs</b>	The outputs from the inspection are: <ul style="list-style-type: none"><li>• the inspection reports and</li><li>• the revised SSTP.</li></ul>

**Task 3: Approve and Distribute SSTP**

<b>Objectives</b>	The purpose of this task is to allow the inspectors to formally approve the SSTP.
<b>Dependencies</b>	This task is dependent on Task 2.
<b>Responsibilities</b>	The test engineer is responsible for obtaining all the required signatures.
<b>Inputs</b>	The input for this task is an inspected and revised SSTP.
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Execution of this task requires completion of the following activities: <ul style="list-style-type: none"><li>• Distribute the revised SSTP to the inspectors</li><li>• Obtain all required signatures</li><li>• Distribute the SSTP</li></ul>
<b>Exit Criteria</b>	Approval has been obtained for the SSTP. The software project manager and project manager receive a copy of the signature sheet.
<b>Outputs</b>	The output from this task is the approved SSTP.
<b>References</b>	None.

**Task 4: Produce Executable Code and Prepare Test Environment**

<b>Objectives</b>	The purpose of this task is to produce executable code and prepare all necessary hardware, software, tools, data sets, and personnel necessary to execute the system tests.
<b>Dependencies</b>	This task is dependent on Task 3.
<b>Responsibilities</b>	The test engineer is responsible for preparing the test environment. A software developer is responsible for building the executable code.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• SSTP</li><li>• Source Code from the Integration Phase</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Execution of this task requires completion of the activities listed below. <ul style="list-style-type: none"><li>• Produce executable code</li><li>• Determine the tests to be run</li><li>• Determine the hardware, tools, and data sets required</li><li>• Schedule the necessary testing personnel</li><li>• Gather the hardware, tools, and data sets required</li><li>• Assemble the SSTP, Test Log, Incident Report, and any other documentation required</li><li>• Install the executable code to be tested</li></ul>
<b>Exit Criteria</b>	The test environment has been prepared.
<b>Output</b>	The output from this task is the test environment.
<b>References</b>	None.

### **Task 5: Execute Tests**

<b>Objectives</b>	The purpose of this task is to: <ul style="list-style-type: none"><li>• execute all the tests described in the SSTP and</li><li>• ensure that the software meets requirements.</li></ul>
<b>Dependencies</b>	This task is dependent on Task 4.
<b>Responsibilities</b>	The test engineer is responsible for executing all tests described in the SSTP. The software project manager reviews the Incident Report and Test Log for completeness. The software developer is responsible for all corrections made to the source code. The surety expert is responsible for ensuring that tests of surety-critical software components are conducted in strict accordance with the software test plans and procedures.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• SDF</li><li>• SSTP</li><li>• Surety Folder</li><li>• Test Environment</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Execution of this task requires completion of the activities listed below. <ul style="list-style-type: none"><li>• During each test session, tests are run according to the procedures described in the SSTP and a Test Log is completed.</li><li>• If reliability metrics are to be used to assess the risk associated with this code, the tester should collect defects data (# of defects, execution time-to-failure, etc.) during test execution. These data are placed into the SDF and may be used to fit a software reliability model and quantify the reliability of this software.</li><li>• After each test session, test results are analyzed, the Incident Report is completed, and the results of surety test cases are added to the Surety Folder.</li><li>• After each test session, if errors were found, the software developer makes all necessary source code corrections.</li><li>• The software project manager reviews the Incident Report and the Test Log for completeness.</li><li>• The surety expert reviews the Incident Report and the Test Log to ensure that test results of surety-critical items are accurately logged, reported, and analyzed. The surety expert will also ensure that deficiencies and discrepancies are corrected and retested.</li></ul>
<b>Exit Criteria</b>	The Incident Report and the Test Log have been reviewed by the software project manager.
<b>Output</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• the Incident Report,</li><li>• the Test Log, and</li><li>• updated Source Code</li></ul>

*Note: The Incident Report and the Test Log are part of the SSTP Template identified in the Project Planning Phase.*

**References**

- MIL-STD-882B [3]
- IEEE 1228 (provides some general guidance about SHA) [10]
- Software Systems Safety Handbook [26]
- A Study on Hazard Analysis in High Integrity Software Standards and Guidelines [27]
- NASA Software Safety Standard [29]

**Task 6: Report and Document Incident Resolution**

<b>Objectives</b>	The purpose of this task is to report all incidents discovered during Task 5 and ensure that all incidents are resolved before continuing with Task 7.
<b>Dependencies</b>	This task is dependent on Task 5.
<b>Responsibilities</b>	The test engineer is responsible for reporting all incidents discovered during test execution and in completion of the incident report. The software project manager and project manager are responsible for overseeing the resolution of the incidents. The surety expert is responsible for updating the surety folder and performing a software change hazard analysis.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• Incident Report</li><li>• MTP</li><li>• Surety Folder</li><li>• Test Log</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Execution of this task requires completion of the activities listed below. <ul style="list-style-type: none"><li>• Incidents are reported according to the procedures described in the MTP.</li><li>• Incidents are evaluated for surety impact and this is documented in the Surety Folder.</li><li>• Incident resolution includes completing a software change hazard analysis.</li><li>• Incident resolution is documented in the Incident Report.</li></ul>
<b>Exit Criteria</b>	The software project manager must approve the Incident Report to verify all incidents have been resolved.
<b>Output</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• Incident Report</li><li>• Software Change Hazard Analysis</li></ul>
<b>References</b>	IEEE 1228 [10]

### **Task 7: Rerun Tests**

<b>Objectives</b>	The purpose of this task is to verify that any changes as a result of incident resolution: <ul style="list-style-type: none"><li>• resolve the incident and</li><li>• do not cause any additional incidents.</li></ul>
<b>Dependencies</b>	This task is dependent on Task 6.
<b>Responsibilities</b>	The test engineer is responsible for executing all tests described in the SSTP. The software developer is responsible for making all source code corrections.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• Incident Report</li><li>• SDF</li><li>• SSTP</li><li>• Surety Folder</li><li>• Test Environment</li><li>• Updated Source Code from Task 5</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start
<b>Task Description</b>	Execution of this task requires completion of the activities listed below. <ul style="list-style-type: none"><li>• Prior to each test session, determine the tests to be run. Associated tests should be run to ensure that additional problems are not caused by the incident resolution.</li><li>• Update test environment by installing new version of executable code.</li><li>• During each test session, tests are run according to the procedures described in the SSTP and a Test Log is completed.</li><li>• If reliability metrics are being used to assess the risk associated with this code, the tester should collect defects data (# of defects, execution time-to-failure, etc.) during test execution. These data are placed into the SDF and may be used to fit a software reliability model and quantify the reliability of this software.</li><li>• After each test session, test results are analyzed, the Incident Report is completed, and the results of surety test cases are added to the Surety Folder.</li><li>• After each test session, if errors were found, the software developer makes all necessary source code corrections.</li><li>• The software project manager reviews the Incident Report and Test Log for completeness.</li></ul>
<b>Exit Criteria</b>	After completion of testing, if any incidents are reported, proceed to Task 6, otherwise proceed to Task 8. The Incident Report and Test Log have been reviewed by the software project manager.
<b>Output</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• the Incident Report,</li><li>• the System Test Log, and</li><li>• Updated Source Code</li></ul>
<b>References</b>	None.

**Task 8: Update SSTP Results Section(s)**

<b>Objectives</b>	The purpose of this task is to update the SSTP as required, incorporating results from test execution.
<b>Dependencies</b>	This task is dependent on Task 7.
<b>Responsibilities</b>	The test engineer is responsible for updating the SSTP.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• Incident Report</li><li>• SSTP</li><li>• Test Log</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Execution of this task requires completion of the activities listed below. <ul style="list-style-type: none"><li>• All defects are resolved and the resolution is documented in the SSTP.</li><li>• The software project manager receives the completed SSTP and verifies the completion of test execution, resolution, and documentation.</li></ul>
<b>Exit Criteria</b>	The SSTP test results section(s) has been completed.
<b>Outputs</b>	The output from this task is an approved SSTP placed under configuration management according to the SCMP.
<b>References</b>	None.

## 4. Closing Remarks

### 4.1.1 Overview

During the course of software development, a number of software products are produced. These products include not only the unit or source code and the executable code, but also numerous documents. Table 4-1 provides a summary of the software products produced by the SDMHCS and the development phase in which each is completed. As software development comes to a close, the disposition of these products must be determined.

**Table 4-1 - Software Products Summary**

<b>Product</b>	<b>Supplier</b>
SDP	Project Planning Phase
MTP	Project Planning Phase
SRS	Requirements Phase
SHA	Requirements Phase
SDD	Design Phase
Integration Test Plan	Design Phase
DHA	Design Phase
Source Code	Implementation Phase
Unit Tests	Implementation Phase
RTM	Integration Phase
Executable Code	Test Phase
SSTP	Test Phase
Surety Folder	Test Phase
SDF	Test Phase
Metrics	Test Phase

### 4.1.2 Completion of System Development

This document has described a software development methodology within the context of a larger system development process. Once software development is complete, the executable code must be integrated by system engineers with other components including hardware and user documentation in order to form a complete HCS. Software developers and test engineers may be involved in integration and test activities at the system level.

Ultimately, the project manager will deliver the completed system to the customer or perhaps to a production facility. In addition to the system itself, delivery may be made of various software products as well as products of the system development. Customer specifications should specify what products are to be delivered. The customer may request delivery of software products in conjunction with system delivery or as the software products are completed.

### 4.1.3 Project Archival and Maintenance

At the conclusion of software development, arrangements should be made for the archival of all software products. This is especially important as long as there are ongoing system development and production activities which may generate modification requests for the software. Even after system development, production, and delivery are complete, there will often exist an ongoing maintenance responsibility. Project archival becomes especially important in this case.

## Closing Remarks

In addition to the archival of software products, archival of the development and testing environments may be necessary in order to support future maintenance efforts. Support software and hardware will need to be archived so that the appropriate environment can be re-created at a later date. Directions on how to assemble testing environments are usually included in the SSTP. Instructions for re-assembling the software development environment are often given in a separate maintenance procedure. See [25] and DI-IPSC-81929 in [2].

### **4.1.4 Verification and Regulatory Agencies**

In the area of HCSs, it is not uncommon for one or more verification or regulatory agencies to have a stake in the software development process. These agencies may request copies of surety-related documentation, reliability metrics, and other software products in order to perform an independent evaluation of the HCS. Such a request for evidence of the software development process may be made known during the Project Planning Phase, in which case it can be fulfilled either at the conclusion of software development or when the requested software product is completed. On the other hand, a request may not be made until deployment of the HCS or even years later when an apparent fault in the HCS is revealed.

### **4.1.5 Metrics**

The metrics which are collected during software development can be utilized in several ways. Metrics which are specific to the software product itself, such as reliability metrics, could be requested as a deliverable to the customer and may be archived along with other project data. Process-related metrics should be considered an organizational asset, and retained regardless of the disposition of other software products. These metrics can be analyzed to determine how the process of software development can be improved for future projects. Lessons learned, collected from all those who contributed to the project, and customer feedback data can also be utilized in any process improvement program.

## 5. Support Processes

### 5.1 Inspection Process

#### 5.1.1 Overview

The Inspection Process is a process that can be initiated by any phase in the SDMHCS. The fundamental objective of software inspections is to improve the quality of software products by recognizing and correcting defects early in the software engineering life cycle.

#### 5.1.2 Inputs and Suppliers

To begin the Inspection Process, a software product that meets all required project standards must be completed. If waivers for standard compliance have been obtained, they must be documented, approved, and available before this process can begin. The inputs to the Inspection Process are summarized in Table 5-1.

**Table 5-1- Inspection Process Inputs and Suppliers**

<b>Input</b>	<b>Supplier</b>
General Objectives	Specified by the SDMHCS phase initiating this process
Inspection Forms	Sandia Guidelines for Software Inspections
SDP	Project Planning Phase
Software Product	Specified by the SDMHCS phase initiating this process
Standards	SDP
Supporting Documentation	Specified by the SDMHCS phase initiating this process
Surety Objectives	Specified by the SDMHCS phase initiating this process
Testing Objectives	Specified by the SDMHCS phase initiating this process
Waivers	Software Project Manager, Customer

### 5.1.3 Outputs and Customers

Table 5-2 contains the outputs from this process.

**Table 5-2 - Inspection Process Outputs and Customers**

<b>Output</b>	<b>Customer</b>
Inspected Software Product	SDMHCS Phase Initiating this process, Quality Assurance
Inspection Management Report	Software Project Manager
Metrics	Quality Assurance, Project Manager, Other Projects

### 5.1.4 Metrics

The process metrics provide a measure of the effectiveness of the Inspection Process and include:

- length of inspections,
- number of major and minor defects identified,
- preparation time for each inspector,
- size of inspection material (e.g., number of pages, lines of code),
- source of defect identified.

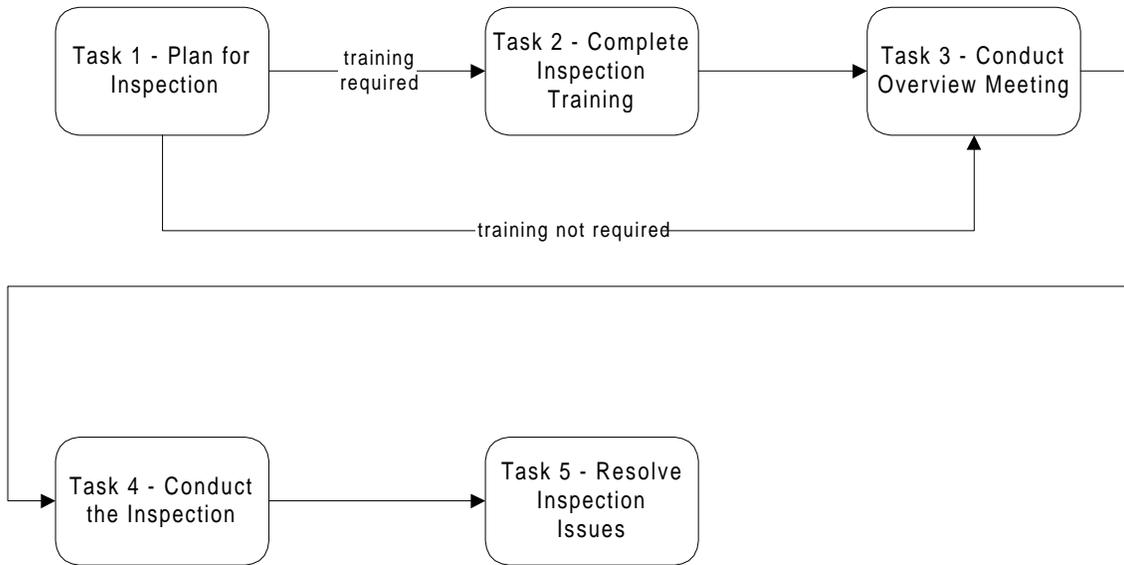
### 5.1.5 Task Summary

The Inspection Process tasks are summarized in the following table. Each of these tasks is described in greater detail in Section 5.1.7.

**Table 5-3 - Inspection Process Task Summary**

<b>Task</b>	<b>Description</b>	<b>Dependencies</b>
1	Plan for Inspection	
2	Complete Inspection Training	1
3	Conduct Inspection Overview Meeting	1, 2
4	Conduct the Inspection	3
5	Resolve Inspection Issues	4

**5.1.6 Task Flow**



**Figure 5-1: Inspection Process Task Flow**

### 5.1.7 Detailed Task Descriptions

This section provides the detailed description of each task listed in Table 5-3. Each task should be completed to successfully carry out the inspection process.

#### **Task 1: Plan for Inspection**

<b>Objectives</b>	The purpose of this task is identify inspection participants, assemble inspection package, and schedule inspection meetings.
<b>Dependencies</b>	Task 1 is dependent on preparation of a software product for inspection.
<b>Responsibilities</b>	<p>The software project manager is responsible for managing all activities required for the completion of Task 1.</p> <p>The author of the software product is responsible for presenting a software product to the software project manager for inspection.</p>
<b>Inputs</b>	<p>The inputs for this task are:</p> <ul style="list-style-type: none"><li>• Product standards</li><li>• Software product</li><li>• Suggestions for inspection participants</li><li>• Waivers, as needed</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start. The software product must meet all applicable standards and inspection criteria identified by the software project manager as required for this type of software product or the software product authors must have documented and approved waivers. Typically, standards and inspection criteria would be documented in the SDP.
<b>Task Description</b>	<p>The author(s) and the software project manager agree on a moderator for the inspection.</p> <p>The author(s), the moderator, and the software project manager identify the necessary inspection participants, including those who require inspection training.</p> <p>The author(s) collects all materials required for a complete inspection package to include inspection material, support materials, and inspection forms.</p> <p>The moderator schedules the overview meeting and all inspection meetings.</p> <p>The moderator distributes the inspection schedule and inspection package to all inspectors.</p>
<b>Exit Criteria</b>	All activities described for this task are completed.
<b>Outputs</b>	<p>The outputs from this task are:</p> <ul style="list-style-type: none"><li>• the inspection package,</li><li>• the inspection schedule, and</li><li>• the list of inspection participants including those requiring inspection training.</li></ul>
<b>References</b>	<ul style="list-style-type: none"><li>• SNL Guidelines for Software Inspections [11]</li><li>• IEEE Std. 1028 [8]</li></ul>

## Support Processes

### **Task 2: Complete Inspection Training**

<b>Objectives</b>	The purpose of this task is to ensure that all inspection participants are trained in the Inspection Process.
<b>Dependencies</b>	This task is dependent on Task 1.
<b>Responsibilities</b>	The software project manager is responsible for managing all activities required for the completion of Task 2.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• inspection training materials</li><li>• list of inspection participants requiring inspection training.</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<ol style="list-style-type: none"><li>1. The software project manager arranges training for inspection participants. Training could be in the form of required viewing of video tapes or an in-house class.</li><li>2. Inspection participant's training records are updated.</li></ol>
<b>Exit Criteria</b>	All activities described for this task are completed.
<b>Outputs</b>	None.
<b>References</b>	None.

**Task 3: Conduct Inspection Overview Meeting**

<b>Objectives</b>	The purpose of this task is to introduce all inspection participants to the software product to be inspected and distribute inspection materials.
<b>Dependencies</b>	Task 3 is dependent on the output from Task 1 and Task 2.
<b>Responsibilities</b>	The moderator is responsible for managing all activities required for the completion of Task 3.  The author is responsible for introducing the software product to be inspected.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• General objectives</li><li>• Inspection package</li><li>• Surety objectives</li><li>• Test objectives</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	<ol style="list-style-type: none"><li>1. The moderator opens the meeting with a brief statement of the subject to be inspected, the inspection objectives, and, if needed, an overview of the Inspection Process.</li><li>2. The author(s) provides a brief tutorial on the software product being inspected, including a summary of any area that might be difficult to understand.</li><li>3. The author provides a copy of the inspection package to each of the participants. The contents of the inspection packet must meet the requirements defined by the project's standard for software inspections.</li></ol>
<b>Exit Criteria</b>	All activities described for this task have been completed.
<b>Outputs</b>	None.
<b>References</b>	<ul style="list-style-type: none"><li>• SNL Guidelines for Software Inspections [11]</li><li>• IEEE Std. 1028 [8]</li></ul>

## Support Processes

### **Task 4: Conduct the Inspection**

<b>Objectives</b>	The purpose of this task is to ensure that the software product is complete, correct, and testable.
<b>Dependencies</b>	This task is dependent on Task 3.
<b>Responsibilities</b>	The moderator is responsible for conducting the meetings. The inspectors are responsible for identifying issues and concerns. The reader is responsible for presenting the software product. The author is responsible for answering questions. The recorder is responsible for recording all issues and concerns.
<b>Inputs</b>	The inputs for this task are: <ul style="list-style-type: none"><li>• General objectives</li><li>• Inspection forms</li><li>• Inspection package</li><li>• Surety objectives</li><li>• Test objectives</li></ul>
<b>Entrance Criteria</b>	The inputs must be available before this task can start. The moderator should also verify that each of the inspectors has prepared adequately.
<b>Task Description</b>	<ol style="list-style-type: none"><li>1. The moderator conducts the inspection according to the approved Inspection Process definition. This includes verifying that all inspection participants are prepared and completing the summary of inspection preparation time. If preparation time is not adequate, the moderator should defer the inspection until all participants are fully prepared.</li><li>2. The recorder collects pertinent data on each defect using the inspection form.</li><li>3. The inspection team discusses all major defects and determines if the software product needs re-inspection</li><li>4. The moderator verifies that all objectives (inspection, test, surety) have been met by the inspection.</li></ol>
<b>Exit Criteria</b>	Each page of the software product has been inspected and all known defects have been documented appropriately..  General, surety, and test objectives have been met by the inspected product.
<b>Outputs</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• the Inspection Summary and</li><li>• the Inspection Defect List.</li></ul>
<b>References</b>	<ul style="list-style-type: none"><li>• SNL Guidelines for Software Inspections [11]</li><li>• IEEE Std. 1028 [8]</li><li>• Any standard required by the software product being inspected</li></ul>

## Support Processes

### **Task 5: Resolve Inspection Issues**

<b>Objectives</b>	The purpose of this task is to resolve open issues and ensure the correctness of resolutions.
<b>Dependencies</b>	This task is dependent on Task 4.
<b>Responsibilities</b>	The author is responsible for resolving inspection issues. The moderator is responsible for ensuring that all issues raised in the inspection are resolved.
<b>Inputs</b>	The input for this task is the Inspection Defect List from the inspection.
<b>Entrance Criteria</b>	The inputs must be available before this task can start.
<b>Task Description</b>	Execution of this task requires completion of the following action items for each item on the Inspection Defect List: <ul style="list-style-type: none"><li>• Investigate defect or open issue</li><li>• Identify a solution</li><li>• Review solution with inspector who raised the issue</li><li>• If the inspector disagrees, seek arbitration from the moderator to resolve the issue</li><li>• If surety-related, record any trade-offs made</li><li>• Close the issue</li></ul>
<b>Exit Criteria</b>	The solutions should be approved by the original inspector(s). The moderator should verify this approval. Verification and approval may require a re-inspection depending on the disposition of the original inspection. The project manager should receive an Inspection Summary Report. All open issues or defects are resolved and approved by the author and the moderator.
<b>Outputs</b>	The outputs from this task are: <ul style="list-style-type: none"><li>• a completed report for management on the inspection, e.g., the Inspection Management Report from Sandia Guidelines for Software Inspections, and</li><li>• an updated software product placed under configuration management according to the SCMP.</li></ul>
<b>References</b>	Sandia Guidelines for Software Inspections [11]

## 5.2 Configuration Management Process

The Software Configuration Management (CM) Process applies administrative and technical procedures throughout the software life cycle to maintain and improve the evolving software product's quality by preventing inconsistencies and defects from being introduced into the software and its documentation as a result of changes. More specifically, it serves to:

- control modifications and releases of the software items,
- control storage, handling and delivery of the software items,
- ensure the completeness, consistency, and correctness of the software items,
- identify, define, and baseline software items in a system, and
- record and report the status of the software items and modification requests.

This process consists of the following activities:

- change control
- configuration evaluation
- configuration identification
- configuration status accounting
- process implementation
- release management and delivery

To be effective, the Software CM Process must be properly managed and well understood by the software development team. Some of the specific activities of CM are:

1. Identification of Configuration Items - The first step in the CM process is to identify those items which will be controlled throughout the software life cycle. Also identified is the specification of the management authority responsible for each item. Configuration identification includes the use of a formal naming convention for each entity. This naming convention often uses a combination of mnemonic labels and version numbers and should be applied to all components of a configuration item.
2. Change Control - The software development process involves a natural progression of change and improvement. The use of software libraries is important to the successful control of software change. The two primary libraries for controlling changes are: personal libraries and project libraries. In the personal library, newly created or modified versions of software entities are maintained by the responsible developer. When a software entity is considered to be stable, it may then be promoted to the project library where it then is accessible to other developers. A mechanism should be established to process change requests from a variety of sources throughout the development process and during operation and support of the system. See [14] for more detail on change control.
3. Status Accounting - The formal process of tracking software entities through the steps in their evolution is referred to as status accounting. Status accounting provides project management with the data necessary to measure and report project progress. The purpose of status accounting is to have the ability, at any point in the development process, to provide the current content of a given software configuration item or Computer Software Component (CSC).
4. Audits and Reviews - Audits and reviews are performed throughout the software life cycle to verify that the software product conforms to the capabilities defined in the specifications or other contractual documentation and that the performance fulfills the customer's requirements. For more detail on audits and reviews, see [14].

Software CM should be integrated into all phases of a software development effort beginning with project planning and ending with testing and the eventual retirement of the product.

In the Project Planning Phase, a Software Configuration Management Plan (SCMP) should be developed for the project which documents all software configuration management activities, schedules, resources, and plan

## Support Processes

maintenance. In addition, the project planning documentation itself (SDP and MTP), should be configured according to the SCMP.

In the Requirements Phase, the CM process should be firmly in place. All requirements documentation, (i.e., SRS, IRS) should be configured according to the SCMP.

In the Design Phase, the software CM system should allow for multiple variations of the design to be tracked and managed effectively. As a design evolves, its history is recorded in the CM system so that a previous state can be returned to, if necessary. In addition, all design documentation (i.e., SDD, IDD) should be configured according to the SCMP.

In the Implementation Phase, software CM management allows for concurrent module development. Several programmers can be assigned to develop modules simultaneously, thereby decreasing implementation time. An individual programmer will have the flexibility to try different approaches to the development of that module without affecting overall implementation. In addition, all software units (CSUs) should be configured according to the SCMP.

In the Integration Phase, the modules that comprise a particular version or revision of the system should be easily identified, even after multiple iterations through the previous phase of the life cycle. In addition, all integrated modules (CSCs) should be configured according to the SCMP.

In the Test Phase, the testing requirements and their associated tests and documentation, should be linked correctly to each of the prospective modules as well as the version or revision they produce. If the system fails formal qualification testing, then the history of each particular module and its associated documentation can be traced back into its development in an effort to locate and isolate defects. In addition, all testing documentation (SSTP) should be configured according to the SCMP.

### 5.3 Software Quality Assurance Process

Software Quality Assurance (SQA) is a planned effort to provide confidence that a software product satisfies the customer's requirements, and meets the user's expectations. In addition SQA is an effort to provide attributes to a project such as portability, efficiency, reusability, and flexibility, while dealing with constraints based on resources and schedule. It is the collection of activities and functions that are used to monitor and control a software project so that specific objectives are achieved with a desired level of confidence.

The SQA Process is a process for providing adequate assurance that the software products and processes in the software life cycle conform to their specified requirements and adhere to their established plans. More specifically, it ensures that:

- the SQA plan and the SDP are compatible,
- an appropriate development methodology is in place,
- the projects use standards and procedures in their work,
- documentation is produced during and not after development,
- documentation is produced to support maintenance and enhancement,
- each software task is satisfactorily completed before the succeeding one is begun,
- mechanisms are in place and used to control changes,
- testing emphasized all the high-risk software product areas, and
- reviews and audits are conducted.

This process may make use of the results of other supporting processes, such as the Inspection Process, and consists of the following activities:

- process implementation
- product assurance
- process assurance
- assurance of quality systems

To be unbiased, the SQA Process needs to be implemented by personnel that are organizationally independent from personnel directly responsible for developing the software product or executing the software development process. This role is typically referred to as the Software Quality Engineer (SQE).

In the Project Planning Phase, the SQE is assigned to the project. The SQE and/or the software project manager are responsible for documenting a Software Quality Assurance Plan (SQAP) which states the methods the project will employ to assure the documents or work products produced and reviewed at each milestone are of high quality. The SQAP should list tools and methodologies, outline project responsibilities, and reference software development guidelines. The SDP and the MTP are reviewed by the SQE for appropriate inclusion of software quality practices and compliance to standards. The SCMP is also reviewed for completeness.

During the Requirements Phase, the following activities are performed by the SQE:

- Audit the SRS for completion and compliance to document standards
- Review SRS
- Review RTM and Surety Folder
- Participate in formal reviews
- Audit the project for compliance to corrective action processes and the SCMP
- Review configuration status accounting reports

During the Design Phase, the following activities are performed:

- Audit the SDD for completion and compliance to document standards

## Support Processes

- Review SDD
- Review RTM and Surety Folder
- Audit preliminary test plans, procedures, and tools (as documented in the SSTP)
- Participate in formal reviews
- Audit the project for compliance to corrective action processes and the SCMP
- Review configuration status accounting reports

During the Implementation Phase, the following activities are performed:

- Audit the completion of unit tests
- Review source code
- Review final test plans and procedures
- Review RTM and Surety Folder
- Audit code for compliance to programming standards
- Audit code and code inspections
- Audit the project for compliance to corrective action processes and the SCMP
- Review configuration status accounting reports

During the Integration Phase, the following activities are performed:

- Audit integration tests
- Review RTM and Surety Folder
- Audit SSTP for compliance to document standards
- Audit the project for compliance to corrective action processes and the SCMP
- Review configuration status accounting reports

During the Test Phase, the following activities are performed:

- Audit the completion of the SSTP (including Incident Reports and System Test Logs)
- Audit system level tests
- Review RTM and Surety Folder
- Audit SSTP for compliance to document standards
- Audit the project for compliance to corrective action processes and the SCMP
- Review configuration status accounting reports

## **6. Conclusions and Recommendations**

### **6.1 Future Enhancements to this Document**

- Formal Reviews
- Requirements Gathering Process
- Risk Management
- User's Manual

## Appendix A - References

- [1] Sandia Preferred Processes for Software Development, Issue 1, February 9, 1994.
- [2] MIL-STD-498, Military Standard for Software Development and Documentation.
- [3] MIL-STD-882B, Notice 1, System Safety Program Requirements.
- [4] IEEE Std. 730-1989, IEEE Standard for Software Quality Assurance Plans.
- [5] IEEE Std. 828-1990, IEEE Standard for Software Configuration Management Plans.
- [6] IEEE Std. 830-1984, IEEE Guide to Software Requirements Specification.
- [7] IEEE Std 1008-1987, Standard for Software Unit Testing.
- [8] IEEE Std. 1028, Standard for Software Reviews and Audits.
- [9] IEEE Std. 1058.1-1987, IEEE Standard for Software Project Management Plans.
- [10] IEEE Std. 1228-1994, IEEE Standard for Software Safety Plans.
- [11] Sandia Software Guidelines for Software Inspections, Draft, Software Quality and Reliability Engineering Department.
- [12] Sandia Software Guidelines, Volume 1, Software Quality Planning, SAND85-2344, August, 1987.
- [13] Sandia Software Guidelines, Volume 2, Documentation, SAND85-2345, December, 1994.
- [14] Sandia Software Guidelines, Volume 4, Configuration Management, SAND85-2347, June 1992.
- [15] Reference for Re-engineering Tool Support, US Air Force, Software Technology Support Center, Hill Air Force Base, May 1992.
- [16] Software Lifecycle Standards for Software Development and Maintenance (Draft Version), dated February 10, 1995, revised May 12, 1995, U.S. Department of Energy.
- [17] Systematic Test and Evaluation Process, An Introduction and Summary Guide, Version 3.0, Dated 2/92, Software Quality Engineering.
- [18] Process Definition for Software Project Planning and Estimating (Preliminary Draft), High Integrity Software Systems Engineering Department 2615, Sandia National Laboratories.
- [19] ISO/IEC 12207:1995, Information technology – Software life cycle processes
- [20] Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems, Version 1.1, Volume 1, February 1995, Department of the Air Force, Software Technology Support Center
- [21] Humphrey, Watts S., *Managing the Software Process*, Addison-Wesley 1990
- [22] Master Test Plan Questionnaire, High Integrity Software Systems Engineering Department 2615, Sandia National Laboratories.
- [23] RTM Tool
- [24] Requisite™ Groupware for Requirements Management Tool, Requisite Inc.
- [25] Process Guidelines for Sandia WR Development, SAND88-0024, July 1992.
- [26] Software Systems Safety Handbook, JPL D-10058, prepared by Jet Propulsion Laboratory for National Aeronautics and Space Administration, May 10, 1993.
- [27] A Study on Hazard Analysis in High Integrity Software Standards and Guidelines, Laura M. Ippolito, Dolores R. Wallace, NISTIR 5589, January 1995.

## Appendix A

- [28] Requirements for the Analysis of Safety Critical Hazards, Interim Defence Standard 00-56 (DRAFT), Ministry of Defence, UK, May 1989.
- [29] (Interim) NASA Software Safety Standard, NSS 1740.13, National Aeronautics and Space Administration, June 1994.

## Appendix B - Definitions

This appendix contains definitions for terms used throughout this document. Definitions related to surety are italicized.

Author - One who creates or generates a software product.

*Accident - An incident with detrimental consequences due to insufficient control of one or more hazards.*

Asset - Any tangible materials of value used in a business operation. Assets include tools, development equipment, and test equipment.

*Assured - Relative confidence or certainty that specific program objectives will be achieved.*

Change Management - See Configuration Management.

Component - One of the parts that make up a system. A component may be hardware or software and may be subdivided into other components.

Configuration Management - A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements.

*Control - The degree to which a system provides positive measures to assure authorized and deny unauthorized use.*

*Criticality - The degree of impact that a requirement, module, error, fault, failure or other item has on the development or operation of a system.*

Customer - The internal or external buyer or recipient of a product.

*Design Hazard Analysis (DHA) - An evaluation of the safety-critical portion of the software design to ensure it correctly implements the safety-critical requirements and introduces no new hazards. For more information about DHA techniques, see reference [3].*

Domain Expert - A source of information, usually from outside the development team, who has expertise in the area from which the project is derived.

*Failure - The state where a system operates outside its required specification.*

*Fault - A defect in a system which may, under certain operational conditions, contribute to a failure.*

Formal Unit Testing - Testing of individual hardware or software units or groups of related units. This testing is done by test engineers and follows formal procedures and is formally documented.

*Hazard - A real or potential condition that can result in unintentional death, injury, occupational illness, or damage to, or loss of, equipment or property.*

*High Consequence System (HCS) - A system whose failure could lead to serious injury, loss of life, destruction of valuable resource (environment, financial data, satellite system), unauthorized use, damaged reputation or loss of credibility, or compromise of protected information.*

*Human Error Probabilities - Probability of a deliberate or inadvertent human action affecting system surety.*

Implementation - The process of converting a software design into executable code.

*Incident - A significant occurrence or event with potentially detrimental consequences.*

Inspectors - Reviewers involved in the Inspection Process.

Informal Unit Testing - Testing of individual hardware or software units or groups of related units. This testing is usually done by the software developer with little or no formal procedures and documentation.

## Appendix B

*Integrity - The degree to which a system or component authenticates and maintains the accuracy and accountability of system data, hardware and software.*

Master Test Plan (MTP) - A set of practices, procedures, and processes defined by software test management for use by the test engineers during the software development effort. The MTP serves as a guide for all testing activities and specifies supporting resource needs.

Methodology - A particular process or set of processes.

Metric - A quantitative measure of the degree to which a system, component, or process possesses a given attribute.

Moderator - One who presides over an inspection.

Module - (1) A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from, an assembler, compiler, linkage editor, or executive routine (2) A logically separable part of a program. Note: the terms "module" and "unit" can be used interchangeably.

*Predictable - Exhibits expected behavior based on prior analysis or experience from test.*

Preliminary Hazard Analysis (PHA) - An analysis performed to identify safety critical areas of a system, provide an initial assessment of hazards, and identify requisite hazard controls.

Priority Assessment - A method of resolving conflicting requirements that uses a predetermined ranking of system characteristics.

Project Management - The processes by which projects are administered from conceptualization through maintenance and retirement.

Project Plan - A document which describes the project management strategies for a project.

Pseudocode - A combination of programming language constructs and natural language used to express a computer program design.

*Quality - (1) The degree to which a system, component, or process meets specified requirements, and (2) The degree to which a system component, or process meets customer or user needs or expectations.*

Quality Assurance - (1) A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements. (2) A set of activities designed to evaluate the process by which products are developed or manufactured.

Rapid Prototyping - A type of prototyping in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process.

*Reliability - The ability of a system or component to perform its required functions under stated conditions for a specified period of time. The system or component also should not function improperly or provide any hidden functionality.*

*Reliability Model - A mathematical construct used to predict, estimate, or assess the reliability of a product.*

*Risk - A measure of the probability of an unwanted occurrence times the magnitude of the consequence.*

Robustness - The degree to which a system or component can function predictably in the presence of invalid or unexpected inputs or stressful environmental conditions.

*Safety - The degree to which the system or component prevents unintended adverse consequences to human life, property, or the environment.*

*Security - The degree to which a system denies unauthorized access in order to protect system assets.*

Software Design Description (SDD) - A representation of software created to facilitate analysis, planning, implementation, and decision making. The software design description is used for communication of software design information, and may be thought of as a blueprint or model of the system.

## Appendix B

Software Development Folder (SDF) - A collection of material pertinent to the development of a given software unit or set of related units. Contents typically include the requirements, design, technical reports, code listings, test plans, test results, problem reports, schedules, and notes for the units.

Software Development Plan (SDP) - A set of practices, procedures, and processes defined by software management for use by team members during the software development effort. The SDP conveys these management controls and a project schedule based on milestones and deliverables.

Software Engineering - The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

Software Engineering Environment (SEE) - The hardware, software, and firmware used to perform a software engineering effort. Typical elements include computer equipment, compilers, assemblers, operating systems, debuggers, simulators, emulators, test tools, documentation tools, and database management systems.

*Software Hazard Analysis (SHA) - An evaluation of software and interface requirements to identify errors and deficiencies that could contribute to a hazard. The SHA should identify design, coding, and test techniques that are encouraged, discouraged, required, and forbidden and is the basis for subsequent software surety analyses. For more information about SHA techniques, see reference [3].*

*Software Reliability - The probability that software will not cause the failure of a system for a specified time under specified conditions. The probability is a function of the inputs to and use of the system as well as a function of the existence of faults in the software. The inputs determine whether existing faults, if any, are encountered.*

Software Requirements Specification (SRS) - Documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces.

Software System Test Plan (SSTP) - A document that provides details of activities required to prepare for and conduct software system test, defines sources of the information used to prepare the plan, defines the software system tests, and defines the test tools and environment needed to conduct the tests.

Standards - Mandatory requirements employed and enforced to prescribe a disciplined and uniform approach to software development. Mandatory conventions and practices are standards.

Statement of Work (SOW) - A document supplied by the customer which defines the work to be performed. All activities, services, and/or products required by the customer, and agreed to by the developer are included in the SOW.

*Surety - Safety, security, control, reliability, and quality.*

*Surety Expert - An individual having special skill or knowledge in the subjects of safety, security, reliability, use control, or quality derived from formal education, research, and/or experience.*

*Surety Folder - Information and records collected as a result of all software surety activities outlined in this document. For more information, see Appendix D.*

System - A collection of components (personnel, procedures, materials, hardware, software) organized to accomplish a specific function or set of functions within a given environment and over a given life span.

Technology Needs - A set of requirements which are based upon the current state of software development practices, procedures, and processes.

Test Level - A decomposition of the test process by execution environment (e.g., unit, integration, system).

Testware - A term analogous to software to reflect all testing related work products (test procedures, test data, test reports, etc.).

*Trusted - Justifiable confidence in the services a computer system delivers.*

Verification - The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

## Appendix C - Overview of Formal Methods

Formal methods are characterized by the use of mathematical techniques to prove correctness. Formal specifications of requirements allow an analysis of the completeness (to some extent) and the consistency of the requirements. Basically this involves applying discrete math such as predicate calculus and set theory to form specifications that can be analyzed for completeness and consistency. State diagrams and Petri nets are often used to express the behavior of the system. These techniques are currently much more accepted in the European software development community than in the United States. An exception to this is the growing use of the cleanroom technology in the United States.

The strength of formal methods is its use in attacking the problems associated with English language specifications. Most defects are inserted at the front end of the software life cycle; however, most defects are found near the end of the software life cycle. This suggests many of the problems are associated with the specification process. Without well-specified system and software requirements, it is especially difficult to analyze the impact of changes to customer requirements -- an analysis that is frequently needed throughout the development life cycle.

There are two important roles in the implementation of a surety-critical system: applications (systems) engineers who understand the functions required of the system as well as the attendant hazards and risks; and software engineers who understand the structuring, implementation, and analysis of software. The communication of system requirements by application engineers, and the derivation of software requirements by software engineers, requires careful coordination and a common understanding of the various levels of specification. This communication is typically hindered by the lack of precision and completeness in the specifications.

More precise specifications are needed. This requires something more suitable to computer science applications than to natural language. Formal (or semi-formal) languages are an appropriate choice. Data flow diagrams, Petri nets, Z (pronounced Zed), and Vienna Development Method (VDM) are all examples of such languages. The use of these languages is an attempt to remove the ambiguity and incompleteness inherent in natural language specifications. In the literature, these languages are instances of "formal methods." Formal methods are used to provide a mathematical approach to expressing system and software requirements and to prove that the actual implementation satisfies the requirements.

It is important to realize that even if formal methods are used to specify requirements and verify that the design and implementation accurately reflect the requirements, there may still be no assurance of system surety. The correctness of the implementation may not specifically address the surety concerns for the system-especially the concern that the system does not do what it is not supposed to do. Such "does not do" surety requirements must also be analyzed so as to be correctly specified. Surety-specific techniques such as hazard identification and analysis must be employed to help ensure that the requirements are adequate to meet the system surety needs.

## Appendix D - Surety Folder Contents

Software surety tasks and activities are interspersed throughout the software development life cycle. The information and records collected as a result of these software surety tasks and activities are compiled in a surety folder. These include:

- Results of hazard analyses
- Information on suspected or known safety problems
- Results of audits performed on surety tasks or activities
- Results of surety testing conducted on all or any part of the entire system,
- Software surety trace matrix

The intent of the surety folder is to convey an awareness of surety concerns to the design team during the software development process, so that, those surety concerns are addressed in the final product. The surety folder acts as a repository for surety-related information. The surety folder evolves during the design process to incorporate historical surety knowledge gleaned from other projects, the results of surety analyses performed during each phase, and findings during the inspections. Specifics about the information added to the surety folder during each phase is provided below.

### Requirements Phase

An initial surety folder is generated upon completion of the software hazard analysis during the requirements phase. The contents of this initial surety folder is a list of the hazards associated with software functions and the criticality of each hazard. The surety folder is then updated during the requirements phase to contain a requirements trace matrix that identifies specific surety requirements that mitigate or control all the hazards in the initial surety folder. This surety requirements trace matrix also incorporates traces from each surety requirement in the SRS to a system-level surety requirement (as documented in the System Specification). Any changes to surety requirements and the rationale for the change during this phase are documented in the surety folder.

### Design Phase

During the design phase, the surety folder's requirements trace matrix is updated to incorporate a trace of each surety requirement to a software entity (function, module, object, etc.) in the design. Undesired states or outputs of the software that impact system surety are associated with each of the surety-critical software entities. Any changes to surety requirements and/or design structure made due to findings from inspections are documented in the surety folder.

### Implementation Phase

Discouraged or forbidden coding practices and historical information on types of coding defects applicable to the specific code to be written are added to the surety folder. Any discrepancy between a surety requirement and the associated code that implements that surety requirement and the resolution of that discrepancy are documented. The surety requirements trace matrix in the surety folder is expanded to provide a description of each surety-critical feature of the code with a reference to the surety requirement(s). Changes from inspections that impact surety are documented.

### Integration Phase

Test cases that demonstrate the compatibility of code involved in surety functions and related code are identified. The results of these test cases are incorporated into the surety folder. Test results that indicate discrepancies between software behavior and surety requirements are documented. Any resulting fixes to the code with the results of regression tests should also be included in the surety folder.

## Test Phase

Test cases that demonstrate the surety features of the software are identified and this information is added to the surety requirements trace matrix. The results from these test cases are also incorporated into the surety folder. Discrepancies, code fixes, and the results of regression tests are documented.

**Distribution:**

<b>Copies</b>	<b>Name</b>	<b>Org.</b>	<b>Mail Stop</b>
10	Lorraine S. Baca	02615	MS0535
5	Larry J. Dalton	02615	MS0535
2	Julie F. Bouchard	02615	MS0535
2	Elmer W. Collins	012335	MS0830
2	Marianna Eisenhour	02612	MS0519
2	David D. Neidigk	02615	MS0535
2	Michael J. Shortencarier	06848	MS1328
2	Patricia A. Trelue	12326	MS0638
2	Danielle M. Clibon	02612	MS0519
2	Mark E. Ekman	12324	MS0491
2	Marie Elena Kidd	02615	MS0535
1	Michael A. Blackledge	12326	MS0638
1	DuWayne A. Branscombe	02612	MS0519
1	Randy D. Dabbs	12326	MS0638
1	Kathy Hiebert-Dodd	05913	MS0573
1	Martha J. Ernest	06523	MS0974
1	Victoria A. Hamilton	06234	MS0449
1	James W. Hole	02671	MS0311
1	Howard R. Kimberly	09672	MS1007
1	Robert J. Longoria	02611	MS0987
1	Gerald McDonald	12326	MS0638
1	William C. Nickell	12300	MS0428
1	David E. Percy	12326	MS0638
1	Tom R. Perea	02665	MS0986
1	John H. Stichman	02600	MS0507
1	Patricia H. Tempel	02615	MS0535

**Central Technical Files**

<b>Copies</b>	<b>Name</b>	<b>Org.</b>	<b>Mail Stop</b>
1	Central Technical Files	08940-2	MS9018

**Technical Library**

<b>Copies</b>	<b>Name</b>	<b>Org.</b>	<b>Mail Stop</b>
5	Technical Library	04916	MS0899

**Review & Approval Desk**

<b>Copies</b>	<b>Name</b>	<b>Org.</b>	<b>Mail Stop</b>
2	Review & Approval Desk for DOE/OSTI	12630	MS0619