

State of Michigan (SOM)

System Maintenance Guidebook (SMG) Version 1.1

**A Companion to the
Systems Engineering Methodology (SEM) of the
State Unified Information Technology Environment (SUITE)**

December 2008



Michigan Department of Information Technology

PREFACE

The initial development of the *System Maintenance Guidebook (SMG)* was published in August 2007, and was developed as part of a continuing effort to improve the quality, performance, and productivity of State of Michigan information systems. Development of the SMG was governed by the Michigan *State Unified Information Technology Environment (SUITE)* initiative. This update incorporates the new System Maintenance Document (SMD) and its related processes.

The purpose of SUITE is to standardize methodologies, procedures, training, and tools for project management and systems development lifecycle management throughout the Michigan Department of Information Technology (MDIT) in order to implement repeatable processes and conduct development activities according to Capability Maturity Model Integrated (CMMI) Level 3 requirements. A formal enterprise level support structure will be created to support, improve and administer all SUITE components, including the System Engineering Methodology (SEM), the SMG, the Project Management Methodology (PMM) and related enterprise initiatives. Until that structure is in place, questions regarding SMG should be sent to the SUITE Core Team at SUITE@michigan.gov.

ACKNOWLEDGEMENTS

The State of Michigan would like to thank the following individuals and organizations that made this version of the State of Michigan System Maintenance Guidebook possible. Without their input and hard work, this would not have been achieved.

INITIAL RELEASE (August 2007)	
Dan Buonodono, Project Management Specialist MDIT Project Management Resource Center	Virginia Hambric, State Division Administrator, MDIT Agency Services – Human Services and MiCSES
Leigh A. Scherzer, Account and Project Manager, MDIT Agency Services – Dedicated Customer Unit, Department of Labor and Economic Growth	Kyle Wilson, IT Specialist, Quality Assurance Team, MDIT Agency Services-Transportation

DECEMBER 2008 RELEASE	
Scott Wager, IT Manager, MDIT Agency Services – Administrative Systems Section, Department of Transportation	Bryan Farr, IT Specialist, Application Development and Support, MDIT Agency Services – Michigan State Police and Department of Military and Veterans Affairs
Celina Kosier, Information Technology Programmer Analyst, MDIT Agency Services – Departments of Attorney General and Corrections	James Suandi, Information Technology Programmer Analyst, MDIT Agency Services, AG and Corrections
Dave Gabler, IT Manager, MDIT Agency Services – Financial Systems Section, Department of Management and Budget	Leigh A. Scherzer, Account and Project Manager, MDIT Agency Services – Dedicated Customer Unit, Department of Labor and Economic Growth

ORGANIZATIONS	
STATE OF MICHIGAN – DEPARTMENT OF INFORMATION TECHNOLOGY	
U.S. DEPARTMENT OF ENERGY – OFFICE OF THE CHIEF INFORMATION OFFICER	

The SMG Development Team owes a large debt to Brenda Coblentz of the U.S. Department of Energy (DOE) for both her encouragement in our efforts and for permitting us the free use of the DOE’s own CMMI Level 3 compliant SEM – Chapter 10 as a basis for this document. In particular, much of this document draws directly from the DOE’s Systems Engineering Methodology, which as of this writing can be found at (http://cio.energy.gov/documents/SEM3_1231.pdf).

Chapter	Page
Chapter: 1.0 Introduction.....	1
Section: 1.1 Background.....	5
Section: 1.2 System Maintenance Definition and Categories.....	6
Section: 1.3 System Maintenance Effort.....	9
Section: 1.4 Call for Projects	11
Chapter: 2.0 System Maintenance	13
Section: 2.1 Initiation and Planning Stage	18
Section: 2.2 Requirements Definition Stage	20
Section: 2.3 Design Stage	23
Section: 2.4 Construction Stage	25
Section: 2.5 Testing Stage.....	27
Section: 2.6 Implementation Stage	29
Chapter: 3.0 Release Management	32
Exhibit 1.0-1 SEM System Maintenance Overview	4
Exhibit 1.4-1 Call for Projects Process Flow	11
Exhibit 2.0-1 Process Model for Maintenance	16
Exhibit 2.1-1 Initiation and Planning Stage	19
Exhibit 2.2-1 Requirements Definition Stage	22
Exhibit 2.3-1 Design Stage.....	24
Exhibit 2.4-1 Construction Stage.....	26
Exhibit 2.5-1 System Testing.....	28
Exhibit 2.5-2 Acceptance Testing	28
Exhibit 2.6-1 Implementation Stage.....	30
Glossary	34

Page inserted for consistency in section start points.

Chapter: 1.0 Introduction**Description:**

The *System Maintenance Guidebook* (SMG) is a set of iterative processes for conducting system maintenance activities. Maintenance practice areas and their subordinate processes prescribe a minimal set of criteria that are necessary for project management and quality assurance processes, control, and management of the planning, execution, and documentation of system maintenance activities.

The use of automated tools to facilitate requirements definition, design, coding, and system documentation is encouraged. The correct selection and implementation of tools varies among the various State of Michigan Department of Information Technology (MDIT) sites and component organizations, and should be coordinated through the MDIT Enterprise Architecture Solution Assessment process.

The State of Michigan has a consistent Project Management Methodology (PMM) in place which can be used for all types of projects.

The SMG is a companion to both the PMM and SUITE. Using these methodologies, staff can move comfortably from applications development, to infrastructure roll out, to software selection to even relocating to new buildings using the same approach throughout the organization.

Significant input for the SMG was obtained from information management programs throughout the country. The SMG integrates State of Michigan and industry best practices and focuses on the quality of system maintenance processes and the work products generated from the processes.

The SMG is derived from the principles and standards advocated by information management industry leaders, such as the Institute of Electrical and Electronics Engineers (IEEE), the Carnegie Mellon University Software Engineering Institute (SEI), the Software Engineering Book of Knowledge (SWEBOK), and the Department of Energy (DOE). ***This methodology is designed to enable State of Michigan project teams to institute and maintain Level 3 maturity on the SEI Capability Maturity Model Integrated (CMMI) process improvement approach.***

The SMG promotes the belief that the result of maintenance is system reliability and the preservation and/or prolonging the life of system assets. The SMG process model does not presume the use of any particular information systems development strategy (e.g., waterfall, spiral). This process is valid regardless of size, complexity, and criticality of the application being maintained.

Project Management:

To the extent possible, all maintenance and operations activities should be managed as a project to gain the benefits inherent in project management and to enable tracking of activities and costs. The extent of project management activity

will vary, and should be tailored according to the size, complexity, and impact of the change or enhancement.

The State of Michigan is adding a System Maintenance process and template to the SUITE methodology in addition to the existing Systems Engineering Methodology (SEM) and SEM Express methodologies. The majority of system maintenance projects are typically smaller in size and will follow the maintenance methodology. Projects with a larger effort would follow SEM Express and SEM. The following guidelines are meant to help MDIT staff determine which process to use and therefore which amount of project documentation and management is required:

Project Size (Effort)	Methodology to Use
0 – 200 hours	System Maintenance Document
201 – 1000 hours	SEM Express
1001 hours or larger	SEM

The above guidelines align with the methodology usage guidelines that are defined in the Project Management Methodology (PMM) document, Chapter 1, pages 1-31 through 1-33.

The following sections provide additional information about the SMG:

- 1.1 Background
- 1.2 System Maintenance Definition and Categories
- 1.3 System Maintenance Effort
- 1.4 Call for Projects

Bibliography:

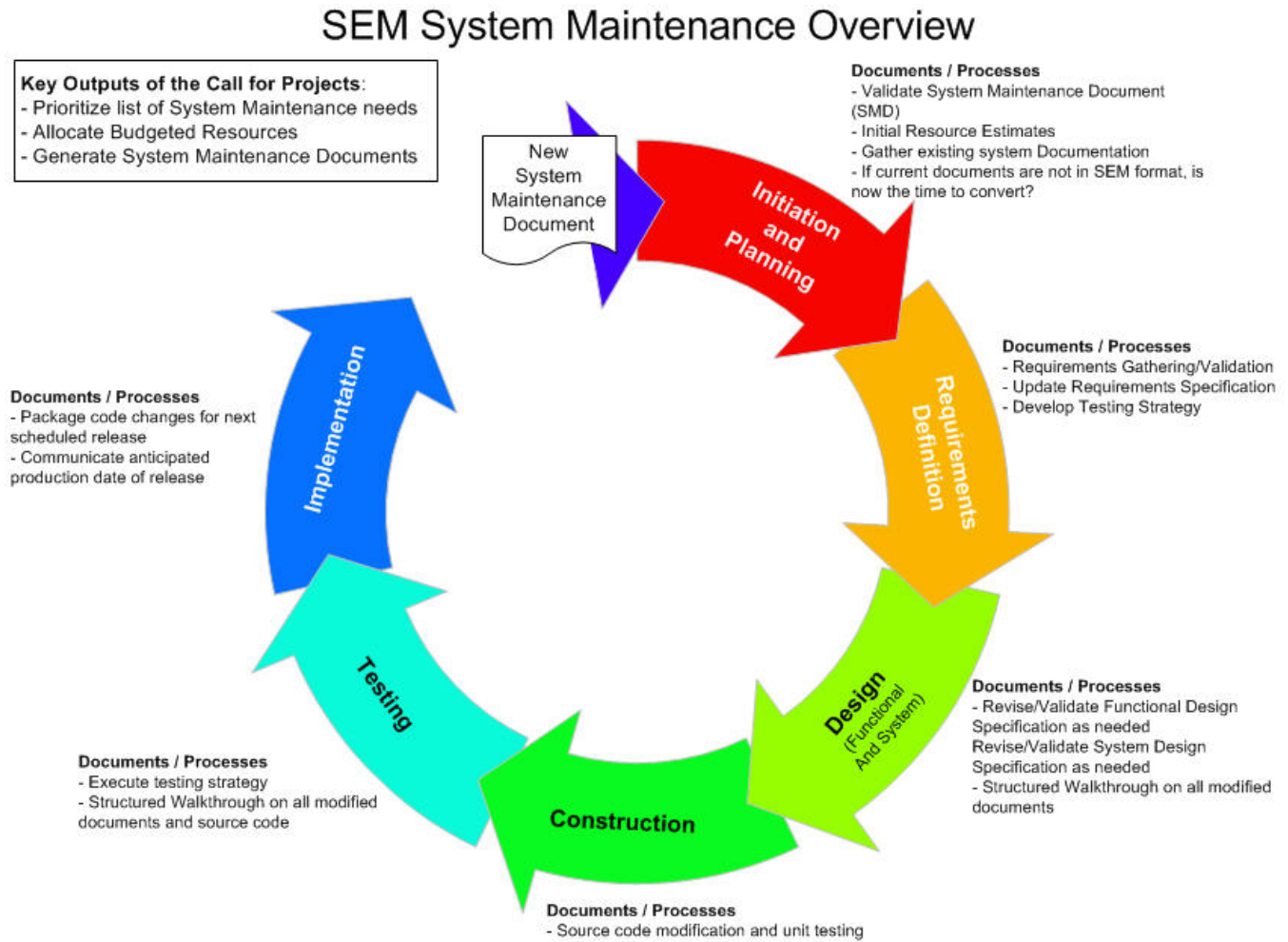
The following materials and web sites were referenced in the original preparation of this guide.

1. State of Michigan Project Management Methodology:
<http://www.michigan.gov/suite>
2. Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY: 1990.
3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Lifecycle Processes*, IEEE Std 1074-1991, New York, 1992.
4. Michigan Department of Information Technology, Agency Services/MDOT *Maintenance and Operations Methodology*.
5. U.S. Department of Energy, *DOE/NV Software Management Plan*, Nevada Operations Office, May 1991.
6. U.S. Department of Energy, *Software Management Guide*, DOE/AD-0028, 1992.
7. Carnegie Mellon University, Software Engineering Institute, *Capability*

Maturity Model: Guidelines for Improving the Software Process, Addison Wesley Longman, Inc., 1994.

8. http://www.inf.ed.ac.uk/teaching/courses/seoc/2006_2007/notes/LectureNote18_SoftwareEvolution.pdf
9. <http://www.sei.cmu.edu/productlines/glossary.html>
10. <http://www.usdoj.gov/jmd/irm/lifecycle/ch11.htm>
11. http://thesource.ofallevil.com/technet/solutionaccelerators/cits/mo/smf/smfisy_sad.mspix
12. http://en.wikipedia.org/wiki/Software_testing
13. http://en.wikipedia.org/wiki/Interface_%28computer_science%29
14. <http://www.compaid.com/caiinternet/ezine/alainapril-maintenancemodel.pdf>
15. <http://onlinepubs.trb.org/onlinepubs/nchrp/optime/optimeusersguide.pdf>
16. http://en.wikipedia.org/wiki/Release_Management

Exhibit 1.0-1 SEM System Maintenance Overview



Section: 1.1 Background***Description:***

Software maintenance is an integral part of any systems engineering lifecycle. In the past, however, it has not received the same degree of attention as the other lifecycle stages. Historically, systems development has had a much higher profile than systems maintenance in most organizations. This is now changing, as organizations strive to squeeze the most out of their systems development investments by keeping software systems operating as long as possible. Concerns about the Year 2000 rollover focused significant attention on the maintenance phase, and the Open Source paradigm has brought further attention to the issue of maintaining software artifacts developed by others.

Broadly speaking, systems maintenance is the totality of activities required to provide cost-effective support to systems. Activities that lay the ground work for system maintenance are performed during all stages of the MDIT Systems Engineering Methodology (SEM).

Maintenance is needed to ensure that the system continues to satisfy user requirements. Maintenance is applicable to systems developed using any systems engineering methodology (e.g., waterfall). Maintenance must be performed in order to:

- Correct faults
- Improve the design
- Adapt systems so that different hardware, software, system features, and telecommunications facilities can be used
- Improve performance
- Maintain operational status

Section: 1.2 System Maintenance Definition and Categories***Maintenance
Definition:***

The Institute of Electrical and Electronics Engineers (IEEE) definition of software maintenance states that:

“Software maintenance is the process of supporting a software product or system after implementation to maintain operational status, correct faults, improve performance or other attributes, or adapt to a changed environment.”

This definition reflects the historic common view that software maintenance is a post-implementation activity: it starts when a system is released to the client or user and encompasses all activities that keep the system operational and meet the user’s needs. This view is well summarized by the classic waterfall models of the systems engineering methodology, which generally comprise a final phase of maintenance and operations.

It is essential, however, to adopt a lifecycle approach to managing and changing software systems, one which looks at all aspects of the development process with an eye toward maintenance. Thomas M. Pigoski, founder and CEO of TechSoft and member of the IEEE panel for development of a single software maintenance standard, captures the need to begin maintenance when development begins in a new, more inclusive definition:

“Software maintenance is the totality of activities required to provide cost-effective support to a software system. Activities are performed during the pre-delivery stage as well as the post-delivery stage. Pre-delivery activities include planning for post-delivery operations, supportability, and logistics determination. Post-delivery activities include software modification, training, and operating a help desk.”

This definition is consistent with the approach to systems taken by the International Organization for Standardization (ISO) in its standard on software life cycle processes. It definitively dispels the image that systems maintenance is all about fixing defects or mistakes. It is also consistent with the MDIT *System Engineering Methodology* (SEM) which introduces maintenance in the Initiation and Planning Stage (Chapter 3, Activity 3.2).

The maintenance of software systems is motivated by a number of factors:

- To provide continuity of service: This entails fixing defects, recovering from failures, and accommodating changes in the operating system and hardware.

- To support mandatory upgrades: These are usually caused by changes in government regulations, and also by attempts of vendors to maintain a competitive edge of rival products. These are not to be mistaken for enhancements.
- To support user requests for certain improvements: An example would be performance tuning.
- To facilitate maintenance work: This usually involves code and database restructuring and updating documentation.

Maintenance Categories:

Maintenance involves activities or costs associated with the ongoing upkeep of the application. Maintenance includes all break/fix requests, optimizing the application, operational support and change management for all requirements identified by MDIT personnel. **Major adaptive changes (e.g. addition of costly new user requirements or porting the system to a new platform) should be carried out as separate new development projects using the SEM.**

Types of maintenance are:

- **Emergency Maintenance (break/fix):** Unscheduled corrective maintenance. While not specifically addressed in the SMG, emergency maintenance is classified into two categories:
 - Production Issues: Issues that stop business operations and must be corrected ASAP. They are performed outside the SMG Release Management process (see Chapter 2 of this guidebook) and involve greater risk due to reduced levels of quality assurance and testing.
 - Urgent Issues: Issues that do not stop business operations, but have a significant impact on them. These issues are corrected on an accelerated basis, using standard SMG processes but outside the Release Management process.
- **Corrective Maintenance:** Identify and remove non-break/fix defects; correct actual errors. These issues are identified and processed according to the SMG and specific system governance in place. They are grouped into planned, scheduled maintenance releases by priority status. Inclusion in the release is determined first by the priority set by the business and secondly by MDIT analysis of resource requirements and dependencies. These are prioritized into two (2) categories:
 - Important: Issues that are identified within the standard governance process, but which are deemed important. Generally

speaking, these issues will be resolved in the earliest scheduled maintenance release possible

- Routine: Issues that are routinely identified and prioritized by the client. These issues then flow into the Release Management process.
- **Perfective Maintenance:** Improves the system without changing its functionality; improves performance, dependability, and maintainability, safety, reliability, efficiency or cost-effectiveness of operation.
- **Adaptive Maintenance:** Modifies the system to keep it up to date with its environment; adapt to a new/upgraded environment by providing new functionality to address requirements that crop up due to changes in the environment (hardware, interfaces, operating system, middleware) or new regulations that impact client operations.
- **Preventive Maintenance:** Identifies and detects latent faults. Changes to the existing system so as to reduce the risk of failure while operating. Preventive maintenance is not specifically addressed in this guide.

Section: 1.3 System Maintenance Effort

Costs: Understanding the *categories* of system maintenance helps to understand the structure of systems maintenance effort. Also, understanding the *factors* that influence the maintainability of a system can help to manage available system maintenance resources.

The current industry standard distribution of maintenance effort across the four system maintenance categories is as follows:

Corrective (approx. 21%)

- 12% emergency
- 9% routine

Adaptive (approx. 25%)

- 18% data environment adaptation
- 7% changes to hardware or operating system

Perfective (approx. 50%)

- 41% enhancements for users
- 6% improve documentation
- 3% other

Preventive (approx. 4%)

- 4% improve code efficiency

System maintenance consumes a major share of system lifecycle financial resources. A common perception of system maintenance is that it merely fixes faults. However, studies and surveys over the years have indicated that almost 80% of the system maintenance effort is used for non-corrective actions. One researcher describes the way in which system maintenance managers often group enhancements and corrections together in their management reports. This inclusion of enhancement requests with problem reports contributes to some of the misconceptions regarding the high cost of corrections.

Factors: Some of the technical and non-technical factors affecting system maintenance efforts are:

- Application type
- System novelty
- System maintenance staff availability
- System life span
- Hardware characteristics
- Quality of system design, construction, documentation and testing

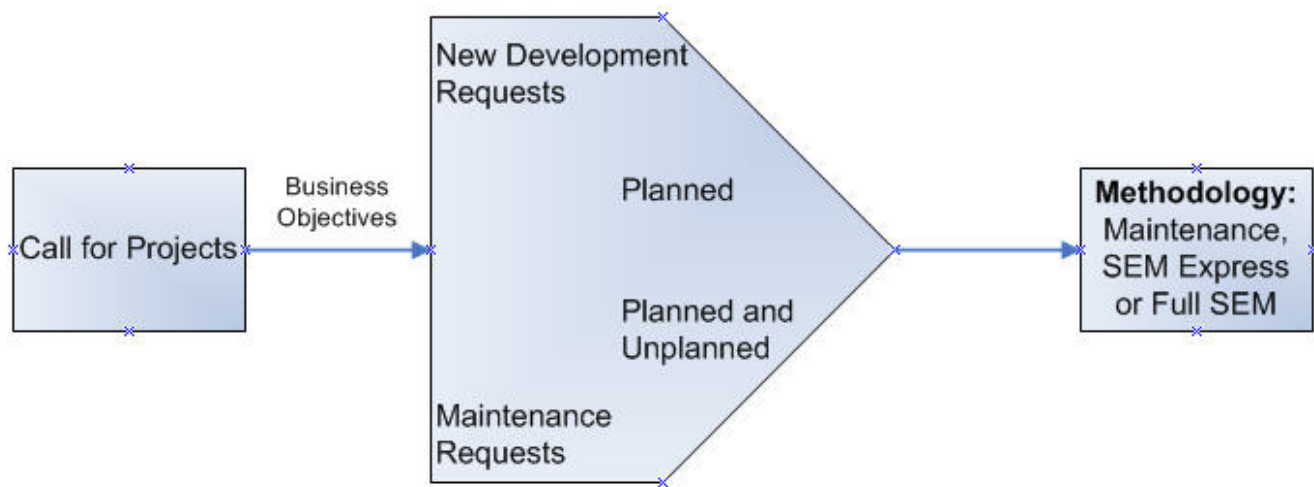
As already noted, many system maintenance activities are similar to those of systems engineering. System maintenance staff perform analysis, design, coding, testing and documentation. They must track requirements in their activities just as is done in development, and update documentation as baselines change.

System maintenance staff may also perform supporting activities, such as system maintenance planning, system configuration management, verification and validation, system quality assurance reviews, audits and user training.

Section: 1.4 Call for Projects**Process:** Call for Projects (quarterly, semi-annually or annually, including maintenance)**Responsibility:** Client Services Director or equivalent, System Maintenance Team**Description:** The Client Services Director (CSD) along with the appropriate system managers and client representatives update strategic plans and identify system engineering business objectives to be addressed in the next fiscal year or portion thereof.

During the Call for Projects, business objectives are identified, categorized, and assigned an initial priority ranking. Each objective is evaluated to determine its classification and handling priority. Maintenance objectives should be identified according to the maintenance types identified in Section 1.2 – *System Maintenance Definition and Categories*. The factors discussed in Section 1.2 should be considered when assigning a priority to system maintenance requests.

The CSD will develop the Maintenance and Operations Strategy (MOS), the Resource Plan and Budget Estimate. These documents will be delivered to the client for review and approval.

Exhibit 1.4-1 Call for Projects Process Flow**Input:** Input to the Call for Projects is one or more business objectives.**Process:** If a modification to a system is required, the following activities must occur within the call for projects process:

- Assign an identification number
- Categorize the type of maintenance

- Analyze the modification to determine whether to accept, reject, or further evaluate
- Prioritize the modification according to the following categories:
 - Mandatory (e.g., legal, safety, payroll)
 - Standard: Issues identified that can be placed into the standard Release Schedule. These have 2 priorities:
 - Important: Has associated benefits; e.g., productivity gains, new business drivers
 - Routine: Nice to have (lower priority)

Control: Business Objectives and process determinations are uniquely identified.

Outputs: The outputs of the Call for Projects are listed below:

- Business Objectives Document (quarterly, semi-annually or annually)
- Initial requirements list
- Initial prioritization and categorization
- Verification data (for corrective modifications)
- Maintenance and Operations Strategy (quarterly, semi-annually or annually)
- Resource Plan (quarterly, semi-annually or annually)
- Budget Estimate (quarterly, semi-annually or annually)

Chapter: 2.0 System Maintenance

Description: The basic maintenance process model includes input, process, output, and control. It is based on the same information systems engineering principles and preferred practices that lower risk and improve quality, as described in the SEM. The Systems Maintenance Guidebook (SMG) process model is founded on the concept that planned changes should be grouped and packaged into scheduled releases that can be managed as individual projects using the new System Maintenance Document (SMD). This proven approach allows the maintenance team to better plan, optimize the use of resources, take advantage of economies of scale, and better control outcome in terms of both schedule and product quality.

Each organization performing system maintenance activities should have a local documented procedure for handling emergency changes that cannot be implemented as part of a scheduled release. Generally, these changes include fixes to correct defects and updates to meet unscheduled business or legal requirements. **For purposes of software configuration management, emergency changes should also be integrated into the next release for full regression testing and documentation updates.**

Stages: The activities performed during maintenance are grouped into logically related segments of work called "stages." The maintenance stages follow the same model as a common Software Development Life Cycle, and are consistent with the SEM Express and full SEM stages. The remainder of this guidebook tailors system maintenance activities as depicted in the SMD. The stages are presented in the sections listed below.

- 2.1 Initiation and Planning
- 2.2 Requirements Definition
- 2.3 Design
- 2.4 Construction
- 2.5 Testing
- 2.6 Implementation

A matrix depicting the maintenance process model is provided in *Exhibit 2.0-1, Process Model for Maintenance*.

Project

Management: To the extent possible, all maintenance activity should be managed as a project to gain the benefits inherent in project management and to enable tracking of activities and costs.

Touch Points: In the event any of the following areas within MDIT need to be involved in system maintenance efforts, it may be necessary to follow portions of the SEM Express or SEM methodology:

- Enterprise Architecture (EA)
- Office of Enterprise Security (OES)
- MDIT Contracts and Procurement
- Infrastructure Services
- E-Michigan

Refer to the respective stages in the full SEM for additional guidance for interaction with these other entities.

Structured Walkthrough (SWT)

Review Process: For small maintenance projects, the SWT processes built into the SMD are sufficient. For those projects in the medium to large categories, the SWT processes established by the SEM should be used.

In each stage (see *Exhibit 2.0-1, Process Model for Maintenance*), one or more structured walkthroughs are conducted to validate work products. The Structured Walkthrough (SWT) is a more formal review and is prescribed by the System Maintenance Document (SMD) for specific project deliverables. SWTs are used to find and remove errors from work products early and efficiently, and to develop a better understanding of defects that might be prevented. They are very effective in identifying design flaws, errors in analysis or requirements definition, and validating the accuracy and completeness of deliverable work products.

SWT's typically require some advance planning activities, a formal procedure for collecting comments, specific roles and responsibilities for participants, and have prescribed follow-up action and reporting procedures. Frequently reviewers include people outside of the developer's immediate peer group.

Responsibility: Project Manager

Work Products: The SWT Meeting Record form (DIT-0187) is available for the reviewers to record errors found prior to the walkthrough session, and for the scribe to record information discussed during the walkthrough. Upon completion, the presenter or author of the work product compiles a SWT Management Summary Report (DIT-0188) and a copy is placed in the Project File.

The State of Michigan guidance document titled *Structured Walkthrough Process Guide* provides a procedure and forms that can be used for SWTs. This document is available on the MDIT SUITE website. (<http://www.michigan.gov/suite>)

Stage Exit

Review Process: Formal Stage Exit Review meetings are not required, but formal business approval is required and handled using the authorization touch points within the SMD.

Metrics: Metrics/measures and associated factors should be collected and reviewed at appropriate intervals. Refer to the “Measurement and Analysis Process Manual” for further details regarding metrics.

Exhibit 2.0-1 Process Model for Maintenance

	Initiation and Planning	Requirements Definition	Design	Construction	Testing	Implementation
Input	<ul style="list-style-type: none"> System Maintenance Document (SMD) 	<ul style="list-style-type: none"> Project/system document Project file information Validated SMD 	<ul style="list-style-type: none"> Project/system document Existing source code Database(s) Analysis stage output 	<ul style="list-style-type: none"> Current source code Product/system document Results of design stage 	<ul style="list-style-type: none"> Update documentation Test Readiness Review Report Integrated system Acceptance test: Plans, Cases, Procedures Updated system Project / Maintenance Plan 	<ul style="list-style-type: none"> Tested/accepted baselined system
Process	<ul style="list-style-type: none"> Assign change number Assign change number Categorize Accept or reject change Preliminary effort estimate 	<ul style="list-style-type: none"> Detailed analysis Re-document, if needed 	<ul style="list-style-type: none"> Revise: Requirements System doc. Module doc Project plan Create test cases 	<ul style="list-style-type: none"> Code Unit test Test Readiness Review Revisit project risk 	<ul style="list-style-type: none"> Functional test Interface testing Interoperability test Regression testing Test Readiness Review User Acceptance Test (UAT) Functional Configuration Audit (FCA) 	<ul style="list-style-type: none"> Physical Configuration Audit (PCA) Install Training User notification Archival system for backup
Output	<ul style="list-style-type: none"> Validated SMD process determinations 	<ul style="list-style-type: none"> Detailed analysis report Updated: Requirements Modification list Test strategy Project plan 	<ul style="list-style-type: none"> Revised: Modification list Detailed analysis Updated: Design baseline Test plans Project plan Constraints/risks 	<ul style="list-style-type: none"> Updated: Design documents Test documents User documents Training materials Project plan Statement of risk Test readiness review report 	<ul style="list-style-type: none"> Tested system Test reports New system baseline Updated project plan Test Readiness review report FCA report 	<ul style="list-style-type: none"> PCA report Version Description Document (VDD)
Review Assure Approve	<ul style="list-style-type: none"> Structured Walkthrough(s) 	<ul style="list-style-type: none"> Structured Walkthrough(s) 	<ul style="list-style-type: none"> Structured Walkthrough(s) 	<ul style="list-style-type: none"> Structured Walkthrough(s) 	<ul style="list-style-type: none"> Structured Walkthrough(s) 	<ul style="list-style-type: none"> Structured Walkthrough(s)
Metrics	<i>See Measurement and Analysis Guidebook for Maintenance metrics</i>					

Page inserted for consistency in section start points.

Section: 2.1 Initiation and Planning Stage**Responsibility:** System Maintenance Team

Description: In this stage, product changes are identified, categorized, and assigned an initial priority ranking by the client. Each request for a modification (i.e., SMD) is evaluated by MDIT to determine its classification and evaluated by the client to determine its handling priority. The classification should be identified from the following types of maintenance.

- Emergency - Unscheduled corrective maintenance required to keep a system operational.
- Corrective - Change to a product after implementation to correct defects.
- Adaptive - Change to a product after implementation to keep it functioning properly in a changed or changing environment.
- Perfective - Change to a product after implementation to improve performance or maintainability.

The need for modifications can be driven by any number of factors, including:

- Report of system malfunction.
- Operational system upgrades and new versions of resident software (e.g., COBOL, Oracle, .Net).

These factors should be considered when assigning a priority to the SMD. *Exhibit 2.1-1 Initiation and Planning Stage* summarizes the input, process, control, and output for the Initiation and Planning Stage of maintenance.

Input: Input to the Initiation and Planning Stage of maintenance is one or more SMDs.

Process: If a modification to the product is required, the following activities must occur within the maintenance process.

- Assign an identification number
- Categorize the type of maintenance
- Analyze the modification to determine whether to accept, reject, or further evaluate
- Prioritize the modification according to the following categories:
 - o Emergency (follow emergency change procedure and integrate into the next scheduled release or block of modifications)
 - o Mandatory (e.g., legal, safety, payroll)
 - o Required (has associated benefits; e.g., productivity gains, new business drivers)
 - o Nice to have (lower priority)

Outputs:

An additional output of this stage is the validated SMD and the following process determinations. Place a copy of all work products in the Project File.

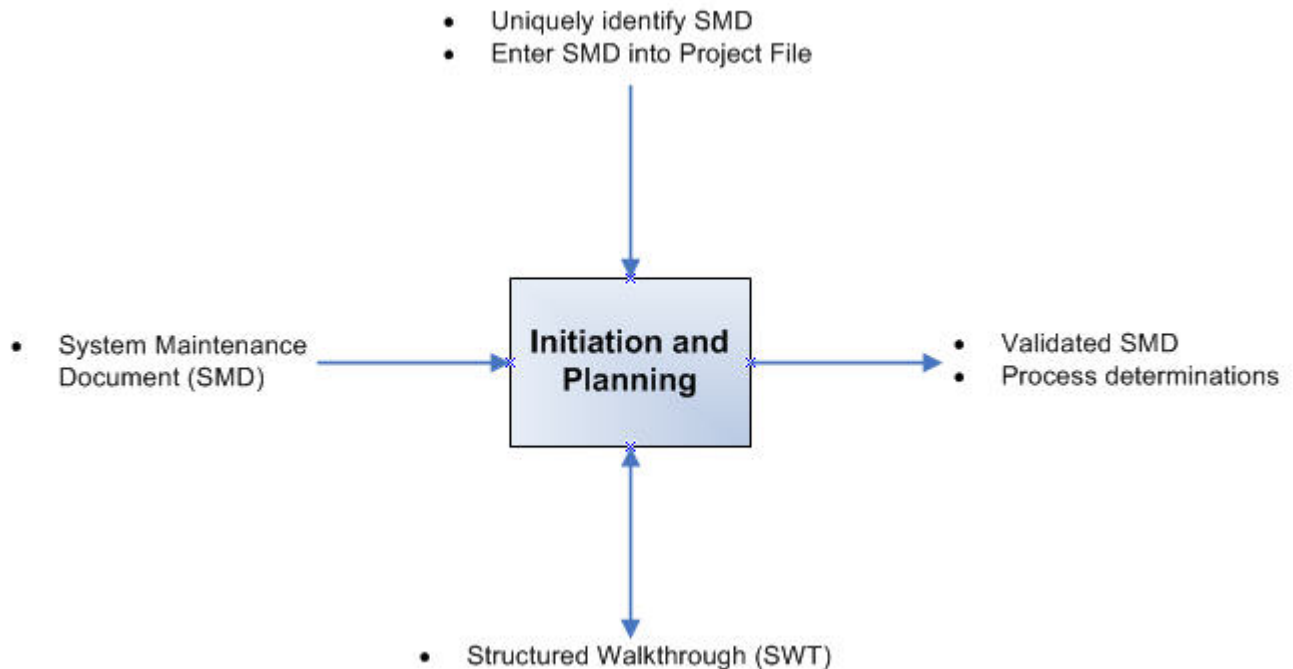
- Statement of the problem or new requirement.
- Problem or requirement evaluation.
- Categorization of the type of maintenance required.
- Initial priority.
- Verification data (for corrective modifications).
- Initial high-level estimate of resources required.
- Assignment to scheduled release.
- Completed “General Information” and “Problem/Enhancement Overview” section of SMD.

Review Processes:

Structured Walkthrough (SWT)

For small maintenance projects, the SWT processes built into the SMD are sufficient. For those projects in the medium to large categories, the SWT processes established by the SEM should be used.

Exhibit 2.1-1 Initiation and Planning Stage



Section: 2.2 Requirements Definition Stage**Responsibility:** Project Team and System Owner

Description: During the Requirements Definition Stage, the Project File information, the System Maintenance Document(s) (SMD) validated in the Initiation and Planning Stage, and the system and project documentation are used to study the feasibility and scope of the modification, and to develop a preliminary Project Plan for design, test, and delivery.

If the documentation is not available or is insufficient and the source code is the only reliable representation of the system, reverse engineering is recommended to ensure the overall integrity of the system. In those cases where long-lived systems have overgrown the initial base system and have poorly updated documentation, *reverse engineering* may be required. The IEEE Standard for Software Maintenance suggests that the process of reverse engineering evolves through the following six steps:

For a smaller scope, or for local analysis on a unit level:

- Dissection of source code into formal units
- Semantic description of formal units and declaration of functional units
- Creation of input/output schematics of units

For a larger scope, or for global analysis on a system level:

- Declaration and semantic description of linear flows
- Declaration and semantic description of system applications (functions grouped)
- Creation of anatomy of the system (system architecture)

Modifications of a similar nature (i.e., affecting the same program(s)) should be grouped together whenever possible, and packaged into releases that are managed as projects. A release cycle should be established and published.

Exhibit 2.2-1 Requirements Definition Stage summarizes the input, process, control, and output for the Requirements Definition Stage.

Input: Input to the Requirements Definition Stage of maintenance includes the following.

- Validated SMD(s)
- Initial resource estimate and associated information
- Project and system documentation, if available

Process: Preliminary analysis activities include the following.

- Make a preliminary estimate of the modification size/magnitude
- Assess the impact of the modification
- Assign the SMD to a block of modifications scheduled for implementation
- Coordinate the modifications with other ongoing maintenance tasks

Once modifications are agreed to, grouped if appropriate, and packaged, analysis progresses and includes the following:

- Define firm requirements for the modification
- Identify elements of the modification
- Identify safety and security issues
- Devise a test and implementation strategy
- Detailed time estimate, fit analysis in release cycle and confirmation of project schedule

In identifying the elements of the modification (creating the preliminary modification list), examine all work products (e.g., system specifications, databases, and documentation) that are affected. Each work product is identified, and generated, if necessary, specifying the portion of the product to be modified, the interfaces affected, the user-noticeable changes expected, the relative degree and kind of experience required to make changes, and the estimated time to complete the modification.

The test strategy is based on input from the previous activity identifying the elements of modification. Requirements for at least three levels of testing, including individual unit tests, integration tests, and user-oriented functional tests are defined. Regression test requirements associated with each of these levels of testing are identified as well. The test cases to be used for testing to establish the test baseline are revalidated.

Control: Control of the Requirements Definition Stage activities includes the following:

- Retrieval of the relevant version of project and system documentation from the configuration control function of the organization
- Review of the proposed changes and engineering analysis to assess technical and economic feasibility, and correctness
- Identification of safety and security issues
- Consideration of the integration of the proposed change within the existing product
- Verification that all appropriate analysis and project documentation is updated and properly controlled
- Verification that the test function of the organization is providing a strategy for testing the change(s), and that the change schedule can support

- the proposed test strategy
- Review of resource estimates and schedules; verification of accuracy
- Technical review to select the problem reports and proposed enhancements to be implemented in the new release

Outputs:

Outputs of the Requirements Definition Stage include the following new or revised artifacts:

- Updated requirements (including traceability list)
- Test strategy
- Complete the requirements area of the “Problem/Enhancement Overview” section of the SMD

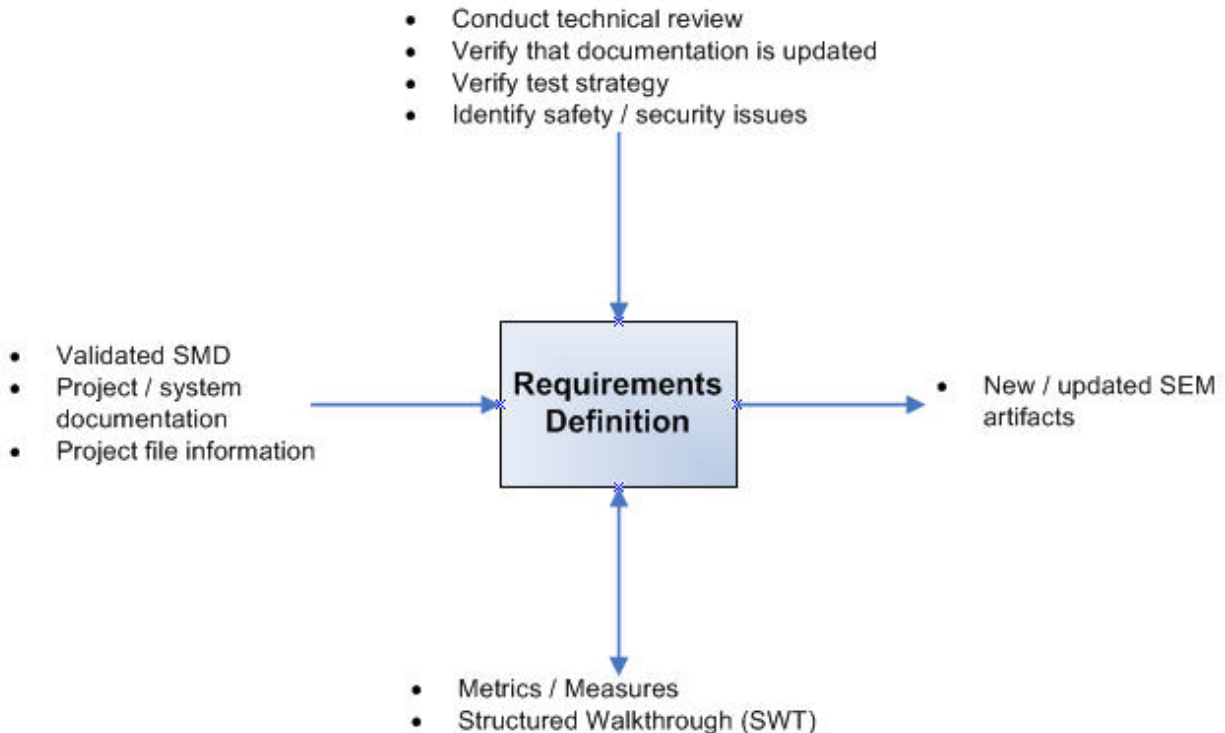
Review Processes:

At the end of the Requirements Definition Stage, a risk analysis is performed. Using the output of the Requirements Definition Stage, the preliminary resource estimate is revised, and a decision is made on whether to proceed to the Design Stage.

Structured Walkthrough (SWT)

For small maintenance projects, the SWT processes built into the SMD are sufficient. For those projects in the medium to large categories, the SWT processes established by the SEM should be used.

Exhibit 2.2-1 Requirements Definition Stage



Section: 2.3 Design Stage

Responsibility: Project Team

Description: In the Design Stage, all current system and project documentation, existing software and databases, and the output of the Requirements Definition Stage are used to design the modification to the system. *Exhibit 2.3-1* summarizes the input, process, and output for the Design Stage of maintenance.

Input: Input to the Design Stage of maintenance includes the following:

- Requirements Definition Stage output, including:
 - o Detailed analysis
 - o Updated statement of requirements
 - o Preliminary modification list (identification of affected elements)
 - o Test strategy
- System and project documentation.
- Existing source code, comments, and databases.

Process: The process steps for the Design Stage include the following:

- Identify selected modules
- Modify module documentation (e.g., data and control flow diagrams, schematics)
- Create test cases for the new design, including safety and security issues
- Identify/create regression tests
- Identify documentation (system/user) update requirements
- Update modification list
- Document any known constraints that influence the design, and any risks that have been identified. Where possible, actions, taken or recommended, that mitigate risk should also be documented.

Control: The following control activities are performed during the Design Stage:

- Verify that the new design is documented as an authorized change
- Verify the inclusion of new design material, including safety and security issues
- Verify that the appropriate test documentation has been updated
- Complete the traceability of the requirements to the design

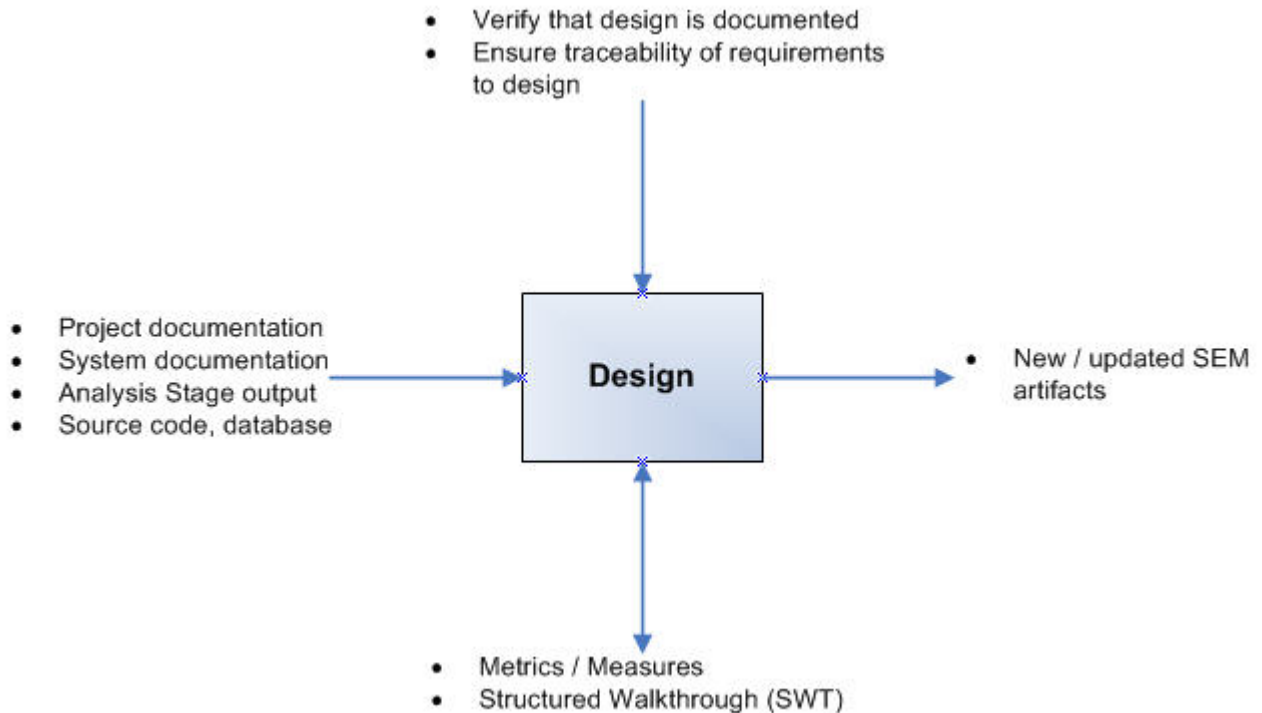
Outputs: Outputs of the Design Stage include the following new or revised artifacts:

- Revised modification list
- Updated design baseline
- Updated test plans

- Revised detailed analysis
- Verified requirements
- Documented constraints and risks
- Completed “Solution and Estimates”, “Design” and “Testing” sections of the SMD

Review Processes: **Structured Walkthrough (SWT)**
For small maintenance projects, the SWT processes built into the SMD are sufficient. For those projects in the medium to large categories, the SWT processes established by the SEM should be used.

Exhibit 2.3-1 Design Stage



Section: 2.4 Construction Stage

Responsibility: Project Team

Description: In the Construction Stage, the results of the Design Stage, the current source code, the project and system documentation, (i.e., the entire system as updated by the prior stages) is used to drive the construction effort. *Exhibit 2.4-1* summarizes the input, control, and output for the Construction stage.

Input: Input to the Construction Stage of maintenance includes the following:

- Results of the Design Stage
- Current source code, comments, and databases
- Project and system documentation

Process: The Construction Stage includes the following tasks, which may be conducted in an incremental, iterative approach:

- Coding and unit testing
- Integration
- Revisit project risk
- Test readiness review

Refer to the SEM for details on coding, unit testing, integration testing and regression testing. The SEM also provides guidance on risk analysis and review, as well as reviewing for test readiness.

Control: Control of the Construction Stage includes the following activities:

- Ensure that unit and integration testing are performed and documented in the Project File
- Ensure that test documentation (e.g., test plans, test cases, and test procedures) are either updated or created
- Identify, document, and resolve any risks exposed during software and test readiness reviews
- Verify that the new product is placed under configuration management control
- Verify that the training and technical documentation have been updated
- Verify the traceability of the design to the code

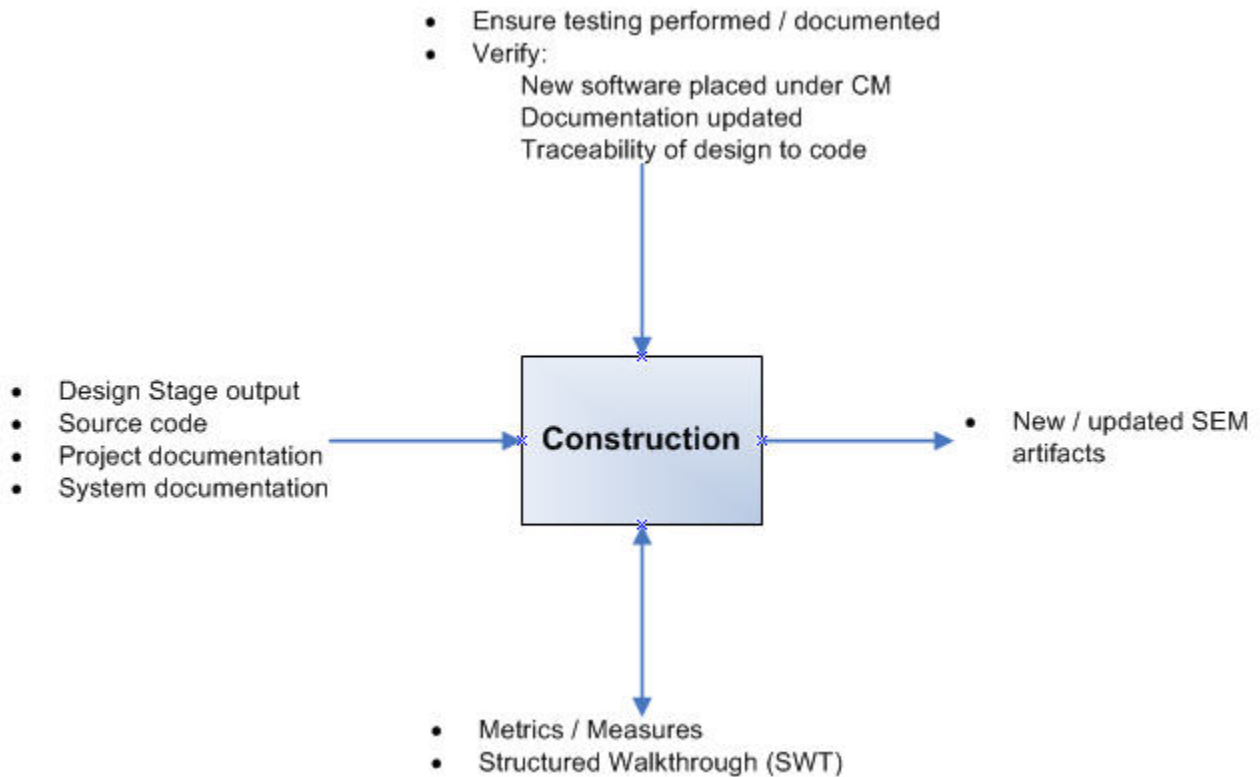
Outputs: Outputs of the Construction Stage includes the following new or revised artifacts:

- Updated system
- Updated design documentation
- Updated test documentation

- Updated user documentation
- Updated training material
- Statement of risk and impact to users
- Test Readiness Review report
- Completed “Unit Test” and “Code Review” sections of the SMD

Review Processes: Structured Walkthrough (SWT)
For small maintenance projects, the SWT processes built into the SMD are sufficient. For those projects in the medium to large categories, the SWT processes established by the SEM should be used.

Exhibit 2.4-1 Construction Stage



Section: 2.5 Testing Stage

Description: Testing is the process used to help identify the correctness, completeness, security, and quality of developed application software. Testing is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. Testing can never completely establish the correctness of arbitrary computer software; testing furnishes a 'criticism' or comparison that compares the state and behaviour of the product against a specification.

An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

Good testing involves much more than just running the program a few times to see whether it works. Testing is the process of executing software in a controlled manner; in order to answer the question "Does this code behave as specified?"

Testing is used in association with verification and validation. **Verification** is the checking of or testing of items, including software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, inspections, and walk-throughs. **Validation** is the process of checking what has been specified is what the user actually wanted.

- Validation: Are we doing the right job?
- Verification: Are we doing the job right?

Activities:

- **System testing:** in which the software is integrated to the overall product and tested to show that all requirements are met (*Exhibit 2.5-1*)
- **Integration testing:** in which the proper interaction between system components is verified
- **Regression testing:** in which earlier successful tests are conducted anew to ensure that changes made in the software have not introduced new defects/side effects
- **User acceptance testing (UAT):** upon which the acceptance of the complete software is based. The clients do this testing. (*Exhibit 2.5-2*)

Outputs:

Outputs of the Construction Stage includes the following new or revised artifacts:

- Complete "System Test", "Integration Test" and "Acceptance Test" sections of the SMD

Review Processes: Structured Walkthrough (SWT)
For small maintenance projects, the SWT processes built into the SMD are sufficient. For those projects in the medium to large categories, the SWT processes established by the SEM should be used.

Exhibit 2.5-1 System Testing

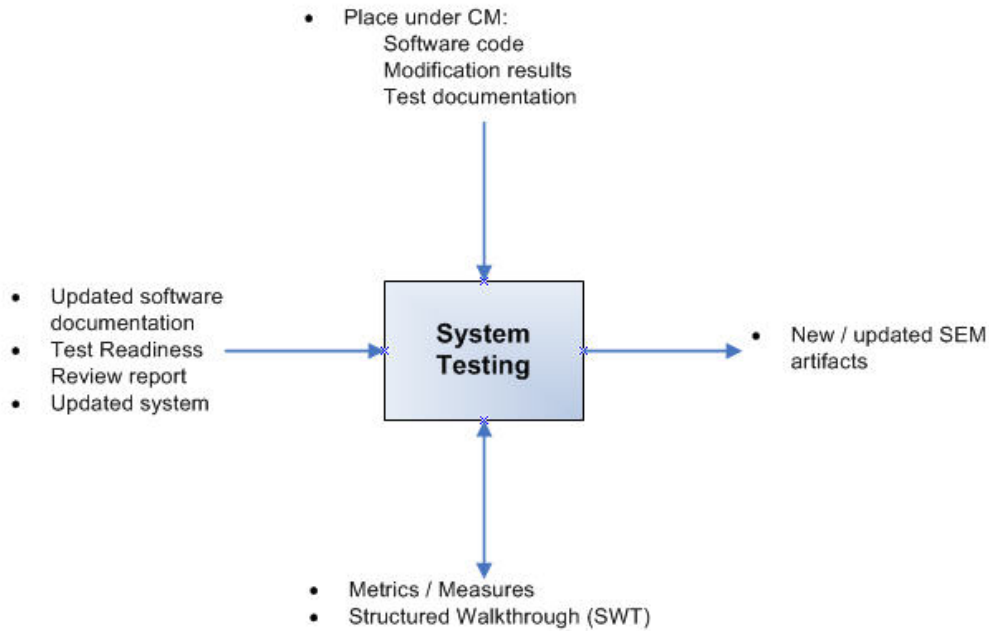
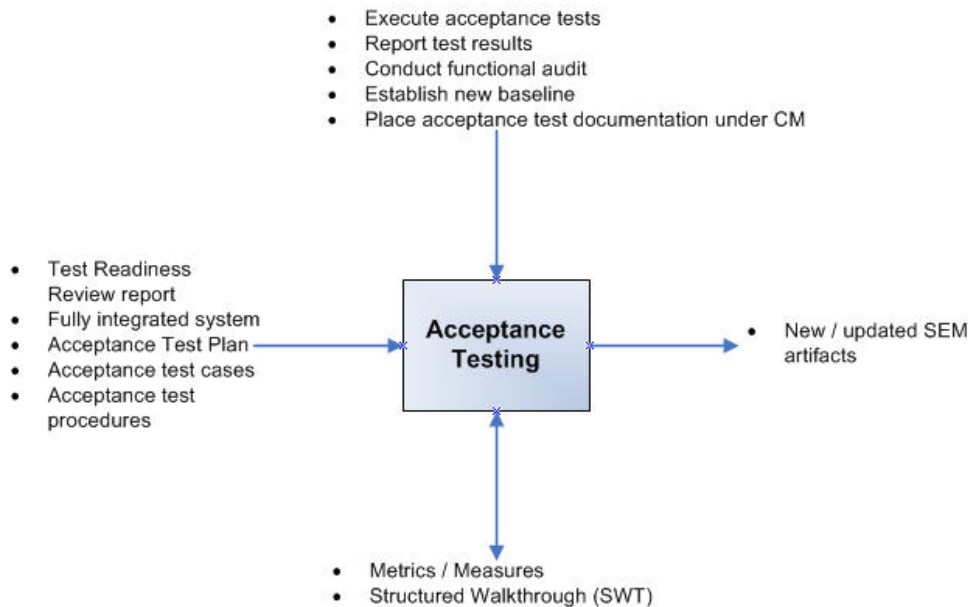


Exhibit 2.5-2 Acceptance Testing



Section: 2.6 Implementation Stage**Responsibility:** Project Team**Description:** This stage describes the requirements for the delivery of a modified system. *Exhibit 2.6-1* summarizes the input, process, control, and output for the Implementation Stage.**Input:** Input to the Implementation Stage of maintenance is the fully tested version of the system as represented in the new baseline.**Process:** The tasks for implementation of a modified system include the following:

- Conduct a Physical Configuration Audit (PCA)
- Notify the user community
- Develop an archival version of the system for backup
- Perform installation and training at the user facility

Control: Control for the Implementation Stage includes the following:

- Arrange and document a Physical Configuration Audit
- Provide access to system materials for users, including replication and distribution
- Complete the version description document
- Place under configuration management control

Outputs: Output of the Implementation Stage includes the following new or revised artifacts:

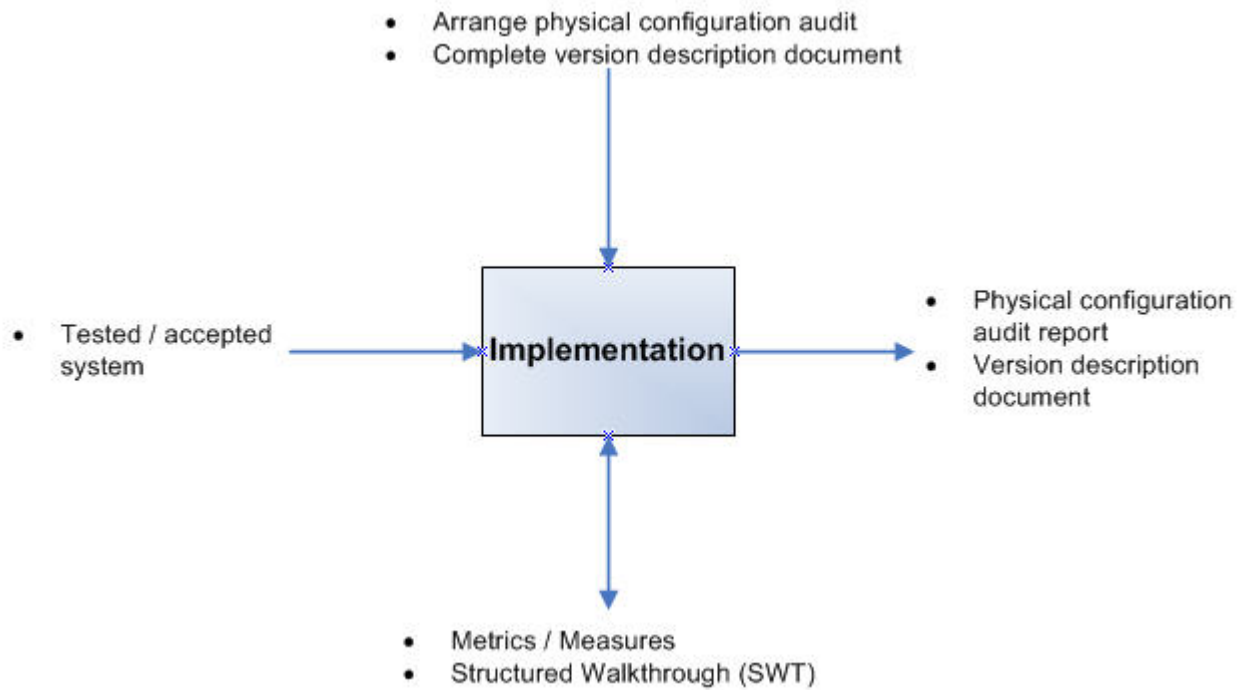
- Physical Configuration Audit report
- Version Control Documentation
- Completed “Authorization to Move to Production” and “Installation” sections of the SMD

The Version control documentation contains information pertinent to the version or release of the system that is being implemented. Information provided includes system name, date delivered, version number, release number, brief description of functionality delivered in the modification, and prerequisite hardware and software with its associated version and release number. This information should be stored in a central location.

Review Processes: **Structured Walkthrough (SWT)**

For small maintenance projects, the SWT processes built into the SMD are sufficient. For those projects in the medium to large categories, the SWT processes established by the SEM should be used.

Exhibit 2.6-1 Implementation Stage



Page inserted for consistency in section start points.

Chapter: 3.0 Release Management

Description: Release Management is the discipline of managing software releases. Releases themselves are a uniquely identified set of files (or perhaps just a single file) that constitute the software release. Binding system maintenance and its related release management activities to existing project management and system engineering processes – as does this guidebook – increases the success rate of release management efforts. This is a best practice for larger applications with multiple concurrent active maintenance activities and may not be needed for smaller applications with less maintenance activity.

The goals of Release Management are:

- Deployment of high-quality release products
- Usage of repeatable, cost-effective process for deploying Releases
- Timely and accurate “building” of Releases

Some of the challenges faced in the Release Management process are:

- Undetected Software Defects
- Outstanding Issues
- Outstanding Risks
- Software Change Requests
- New Development Requests (additional features and functions)
- Packaging and Deployment
- New Development Tasks yet to be completed

Active management of the release process and products provides the project manager (or software release manager) with the following information for use in meeting the above listed challenges:

- What went into the release?
- When was the release deployed?
- Where was the release deployed?
- Why was it deployed?
- How were defects handled when they were reported?

As software systems, software development processes, and related resources become more distributed, they invariably become more specialized and complex. Software products, especially web applications, are typically in an ongoing cycle of development, testing, and release. Add to this the evolution and growing complexity of the platforms on which systems run, it becomes clear there are many moving pieces that must seamlessly fit together to guarantee the success and long-term value of a product or project.

Discussion:

A release is defined according to the needs of the users. This means that:

- Solutions to urgent problems are released as soon as possible only to people experiencing the problem, or who are likely to experience the problem;
- Other changes are released when the users are ready to accommodate them.

Once one or more changes are developed, tested, and packaged into releases for deployment, release management is responsible for introducing these changes into the production IT environment and managing their release. Release management also contributes to the efficient introduction of changes by bundling them into one release and deploying them together.

Refer to the Software Configuration Management template (SEM-0302) for more information.

Glossary

Baseline – (1) An agreed-to description of the attributes of a product, at a point in time, which serves as a basis for defining change. (2) An approved and released document, or a set of documents, each of a specific revision; the purpose of which is to provide a defined basis for managing change. (3) The currently approved and released configuration documentation. (4) A released set of files comprising a software version and associated configuration documentation.

Configuration Audit – Configuration Audits determine to what extent the as-designed/as-tested/as-built product reflects the required physical and functional characteristics specified in the requirements. Functional Configuration Audit (FCA) and Physical Configuration Audit (PCA) are done once, to establish the Product Baseline.

Development Baseline – The baseline comprising the software and associated technical documentation that define the evolving configuration of the system during development. This baseline includes the software design, and implemented code, database schema, and COTS products, and evolves into the product baseline.

Fix number – An indicator of small updates that are to be built into a regular modification or release at a later time. The version, release, modification, and fix levels together comprise the program level (or version) of a program.

Functional Baseline – The baseline comprising documentation and possibly models that specify system functional, data, and technical requirements.

Functional Configuration Audit (FCA) – An inspection to determine whether the (software) configuration item satisfies the functions defined in specifications. Consists of someone acknowledging having inspected or listed each item to determine it satisfies the functions defined in specifications.

Iteration – A distinct sequence of activities with a baselined plan and valuation criteria resulting in a release.

Modification number – The modification level of a program, which is an indicator of changes that do not affect the external interface of the program. The version, release, modification, and fix levels together comprise the program level (version) of a program.

Physical Configuration Audit (PCA) – The formal examination of the "as-built" configuration of a configuration item against its technical documentation to establish or verify the configuration item's product baseline.

Program level – The version, release, modification, and fix levels of a program.

Regression Testing – The process of running a composite of comprehensive test cases that are always run after system modifications to detect faults introduced by modification.

Release number – An indicator of changes to the external programming interface of the program. The version, release, modification, and fix levels together comprise the program level (version) of a program.

Specification – A document that explicitly states essential technical attributes/requirements for a product and procedures to determine that the product's performance meets its requirements/attributes.

Version – (1) One of several sequentially created configurations of a data/document product. (2) A supplementary identifier used to distinguish a changed body or set of computer-based data (software) from the previous configuration with the same primary identifier. Version identifiers are usually associated with data (such as files, databases and software) used by, or maintained in, computers.

Version Description Document (VDD) – A document associated with a product release that describes the version released and describes the versions of the items included in the product.

Version Number – An indicator of the hardware and basic operating system upon which the program operates. The version, release, modification, and fix levels together comprise the program level (version) of a program.