State of Michigan (SOM)

Testing Process Manual Version 1.0

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

March 2010

Michigan Department of Information Technology

PREFACE

This initial development of the *Testing Process Manual* was published in *March 2010*, 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 *Testing Process Manual* was governed by the Michigan *State Unified Information Technology Environment* (SUITE) initiative.

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). SUITE permits MDIT 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 Systems Engineering Methodology (SEM), the Project Management Methodology (PMM), and related enterprise initiatives. Until that structure is in place, questions regarding Testing 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 State of Michigan *Testing Process Manual* possible. Without their input and hard work, this document would not have been created. In addition, we'd like to acknowledge the contributions of various members of other SUITE Workgroups and State testing resources who improved the quality of this document through their participation.

INITIAL RELEASE (March 2010)	
Amy Montgomery, QA/Testing Specialist MDIT Agency Support Services – MDOT/CS/DMB	Sue Tomes, Department Manager, MDIT Agency Services – Human Services
Jan Miller, Testing Senior Analyst, Agency MDIT Support Services – MDOT/CS/DMB	Patty Whitlock, Department Analyst, MDIT Agency Services – Human Services
Sara Podleski, Testing Junior Analyst, Agency MDIT Support Services – MDOT/CS/DMB	Sharon Lewis, Project Manager, MDIT Agency Services – Human Services

ORGANIZATIONS
STATE OF MICHIGAN – DEPARTMENT OF INFORMATION TECHNOLOGY

Document Revisions

The following information is used to control and track modifications made to this document.

Revision Date	Section(s)	Summary
3/26/2010		**Initial document release**

Chapters Page

Exhibits Page

Chapter: **1.0 Introduction**

Section: **1.1 Scope of this Process Manual**

Description: This document is intended to help project staff identify the Who, What, Where, When, Why and How of testing. The information should help teams know where test planning, design and execution information is to be captured as they progress through each SEM Stage.

Notes: The information within this document is based on a full SUITE implementation and should be adjusted accordingly, if using SUITE Express or a tailored approach.

One definition of Quality states, "Quality is the process of continual improvement." For each quality issue, problem or success that occurs throughout the project, details should be documented in a timely fashion and provided to the Project Manager (PM). The PM should then record that information within the Project Lessons Learned document (PMM-18). It is most effective to discuss lessons learned after each stage of the project. Information should be adjusted in the Project Lessons Learned document, as the project progresses.

Section: **1.2 SEM Stages Covered**

Description: This process manual covers the following SEM Stages:

- Initiation and Planning
- Requirements Definition
- Functional Design
- System Design
- Construction
- Testing
- Implementation

Section: **1.3 Purpose of Testing**

Description: In CMMI, testing is a form of validation and verification activities. Validation demonstrates that the product, as provided, will fulfill its intended use. Verification addresses whether the work product properly reflects the specified requirements.

Validation activities use approaches similar to verification (e.g., test, analysis, inspection, demonstration, or simulation). Often, the end-users and other relevant stakeholders are involved. Both validation and verification activities can run concurrently and may use portions of the same environment.

Verification is inherently an incremental process because it occurs throughout the development of the product and should be applied to all work products. The Verification process involves the following: preparation, performance, and identification of corrective action. Verification of work products substantially increases the likelihood that the product will meet the customer, product, and product-component requirements. Peer reviews are an important part of verification and are a proven mechanism for effective defect removal.

Summary: Verification answers the question: "Are we building the right product?" Validation answers the question: "Are we building the product right?"

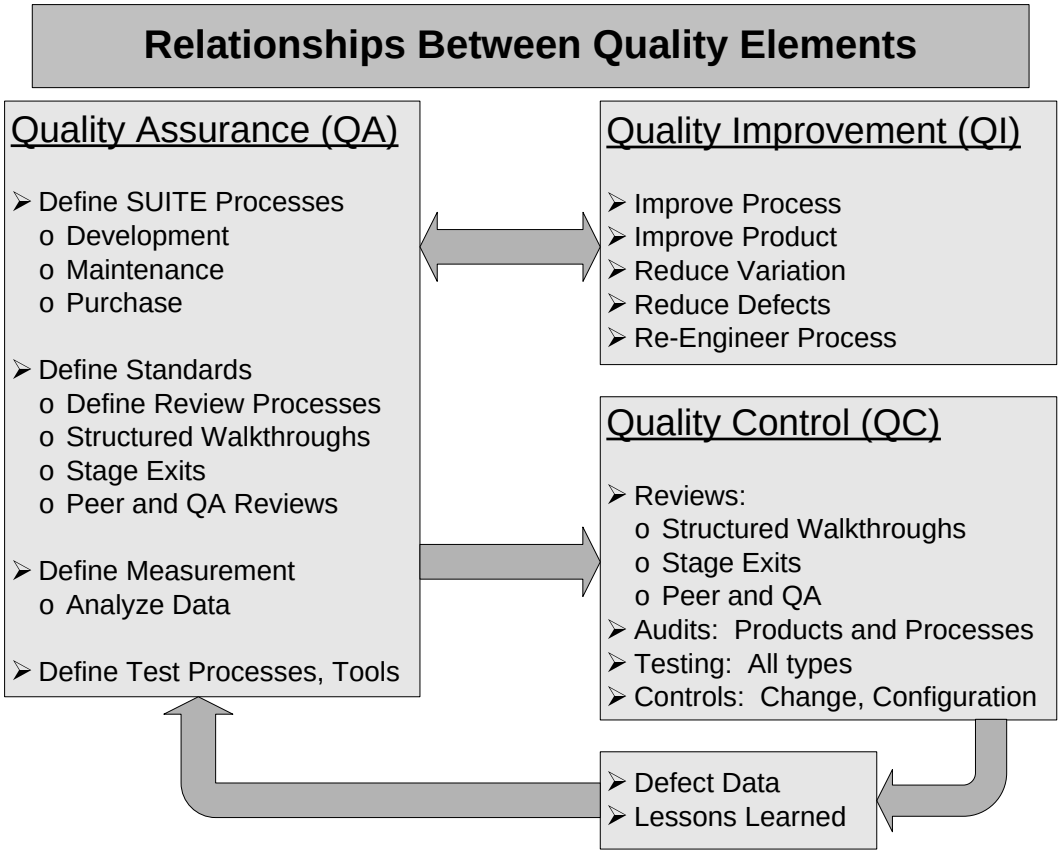
Section: **1.4 Quality Fundamentals**

Description: One common definition of Quality is Dr. Juran's "Fit for use". Another widely-accepted definition is Dr. W. Edwards Deming's "Quality is the Continuous Improvement of All Processes". Together, these two definitions will keep projects on the correct path to provide customers with quality products, services and information.

People frequently use the term Quality Assurance (QA) as a generic term when they actually mean Quality Control (QC). QA (validation) is the process to prevent defects and QC (verification) is the process to find (and fix) defects.

Testing and review activities are QC activities and it is important to understand how all the quality elements fit together.

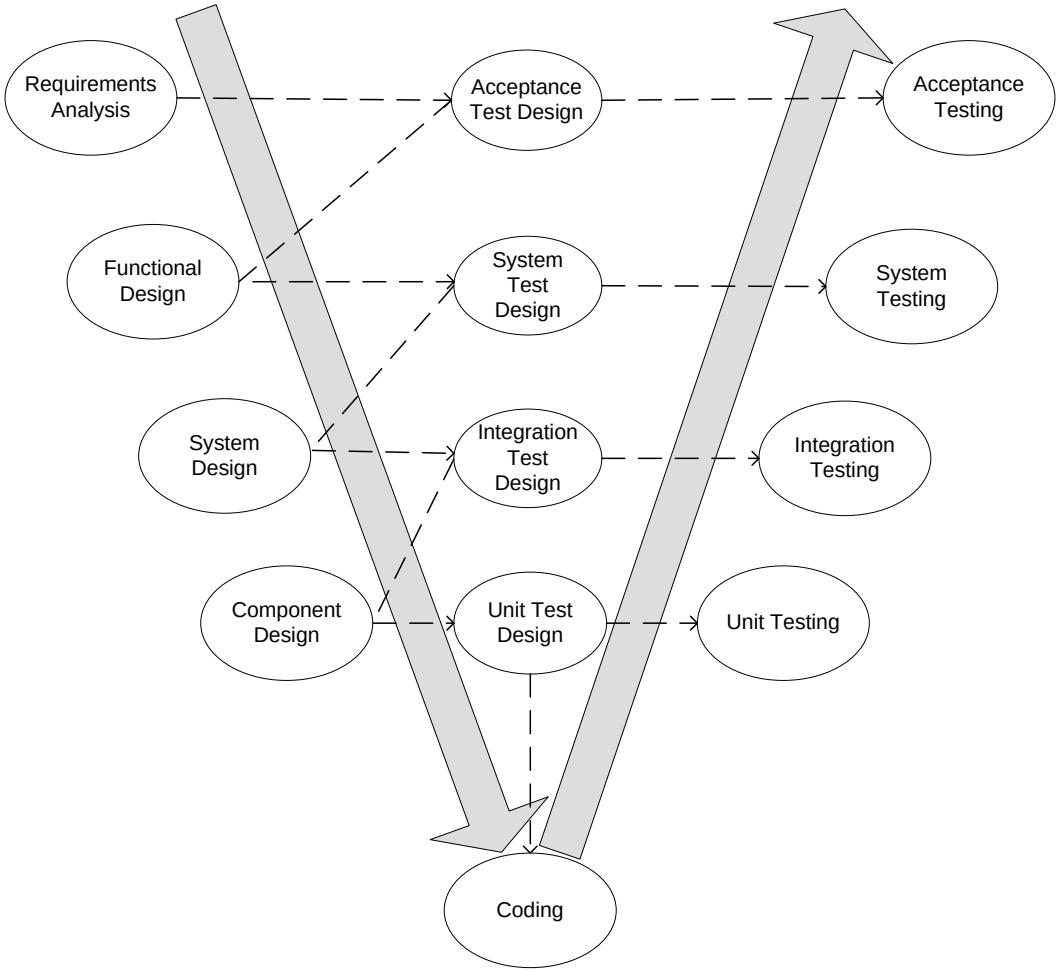
Exhibit 1.4-1 Relationships Between Quality Elements



Section: **1.5 Testing Methodologies**

Description: The traditional and most structured testing model is the V-model. It is based on a sequential waterfall development methodology. The V-model demonstrates the relationships between each stage of the development lifecycle and its associated testing stage.

Exhibit 1.5-1 V-Model Structured Testing Model



Implementation of the V-model is not limited to the waterfall development methodology. It can also work with an iterative development approach (progress through the V can occur quickly as each iteration is being developed and completed).

A simplified version of the V-model can be used when using the Agile Methodology. The left branch (top down) contains Analysis and Design. The bottom of the "V" lists Coding. The right branch (from the base of the "V" upward) lists Testing and Acceptance. User stories (Analysis) are created, followed by the system test (Design) and unit test cases. Development (Coding) is performed, unit test cases are executed (Testing) and User Acceptance Testing (Acceptance) is conducted.

Section: **1.6 Test Types**

Description: There are many different test types. Identification of who should perform each test type and when they should occur is noted in the table below. Other test activities can occur above-and-beyond the core types listed.

Who	Test Type	Definition	SEM Stage(s)
Developer	Unit Test	Confirms that the program logic within an application module produces the expected output when given a known input. Written from a developer's perspective, this ensures that the smallest testable module of an application successfully performs a specific task(s). Unit tests tell a developer that the code is *working properly*.	Construction
Developer	Function Test	Confirms that the logically-grouped modules function according to specifications. Developers write this test from a user's perspective. Testing is based on output only, without any knowledge of internal code or logic. Function tests tell a developer that the code is *working properly*.	Construction
Technical Development Team	Integration Test	Verifies the system components are integrated and working as an application. The technical development team performs this test to uncover errors that occur in the interactions and interfaces between components.	Construction and Testing

Who	Test Type	Definition	SEM Stage(s)
Any Testing Stakeholder	Verification Test	Verifies that a product or product component fulfills its intended use when placed in its intended environment. Any testing stakeholder can conduct this test.	Construction, Testing, and/or Implementation
Technical Team	Performance Test	Measures software performance of batch data, under actual or anticipated volume, as well as on-line transaction response times. Executed by the technical team, this test verifies performance requirements, throughput, and growth capacity. Performance tuning continues throughout the system lifecycle.	Construction, Testing, and/or Implementation
Technical Team	System & Standards Test	Verifies that functional business requirements, business processes, data flows, and other system criteria are met. The technical team tests specific end-to-end business processes until the complete application environment mimics real world use. Verify that Federal, State of Michigan, and department standards are met.	Testing
End-User	User Acceptance Test (UAT)	Validates that the system, as a whole, meets mutually agreed-upon requirements. UAT is completed by end-users of the application and occurs before a client or customer accepts the new system.	Testing

Who	Test Type	Definition	SEM Stage(s)
Any Testing Stakeholder	Regression Test	Re-execution of specific test cases to ensure defects are fixed, to find new defects that may have been introduced, and to confirm module(s) are functioning properly. Any testing stakeholder can conduct this test.	Construction, Testing and Implementation

Section: **1.7 Testing and the Development Lifecycle**

Description: By definition, Testing is a process that assesses the quality of a system. It provides services and deliverables that help the organization manage the risks to system quality. Organizationally, a test team is a group that provides testing services and products for the projects they serve.

It is important for the Project Manager (PM) and project team to understand the value of the services and information that the test team can provide in order to reduce overall risk of project failure. The test team contributes as follows:

1. Identifies potential defects by conducting peer reviews throughout the Requirements and Design Stages.
2. Guides the project with timely, accurate and credible information.
3. Runs tests and captures actual results against expected outcomes.
4. Identifies, reports and re-tests defects.
5. Provides input into the Go/No-Go decision.

In addition to the above, the test team can provide input into the quality risk analysis activity.

In general, the earlier a quality activity can occur in the development lifecycle, the better. Early employment of quality activity is worth time and effort because it results in saving money. The later in the development lifecycle defects are detected and fixed, the more expensive they are to fix, potentially increasing a project's budget and timeline.

Chapter: **2.0 Initiation and Planning Stage**

Key Documents: Initiation and Planning Stage documents related to testing:

- Project Charter (PMM-02)
- Project Plan (PMM-03)
- Quality Plan (PMM-07)
- Project Security Plan and Assessment (DIT-0170)
- Communications Plan (PMM-08)
- Resource Plan (PMM-05)
- Work Breakdown Structure (WBS) (PMM-04)

Document: **Project Charter (PMM-02)**

Tasks: Identify testing stakeholders including, but not limited to, the following:

- Test Lead
- Quality Assurance Resource

Identify roles and responsibilities.

Notes: The test lead and quality assurance resource, along with any other resources deemed necessary, must be consulted throughout all stages of a project.

Document: **Project Plan (PMM-03)**

Tasks: Identify testing stakeholders and points of contact including, but not limited to, the following:

- Test Lead
- Quality Assurance Resource
- Business User Acceptance Test Lead

Document: **Quality Plan (PMM-07)**

Description: The Quality Plan describes what will be done, how it will be reviewed and how it will be accepted.

Tasks: Add the following deliverables to Section C. Deliverable Description:

- Test Plan (SEM-0602)
- Test Type Approach and Report (SEM-0603)
- Test Case (SEM-0606 or equivalent)
- Defect Tracking Log (SEM-0186 or equivalent)

Address the following items in Section D. Acceptance Criteria:

- Review criteria per SUITE template(s) (e.g., Product Process Quality Assurance (PPQA) – PPQA checklists)
- Process for reviews (i.e., Structured Walkthrough)
- Process for Stage Exit(s)

Identify what testing metrics will be used in Section E. Quality Assurance Activities.

Examples: **Types of Testing Metrics:**

- Test cases planned for execution as opposed to test cases that were actually executed. It is good to acknowledge this when the numbers are different and to explain why.
- Final number of defects (grouped by type and priority). This is good summary information to identify where the majority of the defects occurred.
- Defect Acceptance Ratio (%) – This is the number of valid defects identified during test execution.
  [(Number of Valid Defects) divided by (Total Number of Defects) multiplied by 100]%
- Defect Detection Effectiveness (%) – Total Number of Defects Reported includes defects reported by any party other than the test team, including post-delivery defects.
  [(Number of Defects Reported by Test Team) divided by (Total Number of Defects Reported) multiplied by 100]%
- Leaked Defects – Review defects at the end of each stage, starting with Functional Design. See what defects were identified in that stage and whether they should have been detected/resolved in a previous stage. Identification of a root cause, including why or how the defects were leaked, can be valuable feedback into process improvement.
- Defect Rejection Rate (%) – How many defects were rejected? This metric might indicate that the testers need more training before testing begins or that they need to be involved earlier in the process.
  [(Number of Defects Rejected) divided by (Total Number of Defects) multiplied by100]%
- Bad Fix Defect (%) – How many defects were fixed, re-opened and sent back to development to be fixed again? This takes time (and money) and may indicate a problem in the process that can be changed.
  [(Number of Bad Fix Defects) divided by (Total Number of Valid Defects) multiplied by 100]%

Identify how the above metrics will be monitored in Section F. Project Monitoring and Control.

Identify who will be responsible for the QA activities in Section G. Project Team Quality Responsibilities.

Document: **Project Security Plan and Assessment (DIT-0170)**

Tasks: If applicable, include security access for testing tools and environments.

Document: **Communications Plan (PMM-08)**

Tasks: Include the identified testing and QA stakeholders listed in the Project Charter.

Include the methods (e.g., automated tools, Excel) in Section C. Information Type that will be used to inform stakeholders on testing status and defects.

Ensure that Section E. Length of Involvement indicates which stages or time duration the testing and QA resources will be involved. Also ensure that this information matches what was indicated in Section H. Roles and Responsibilities of the Project Charter and Section D. Resource Staffing Plan of the Resource Plan.

Document: **Resource Plan (PMM-05)**

Tasks: Estimate how many testing and QA resources will be needed and for what duration.

Document: **Work Breakdown Structure (WBS) (PMM-04)**

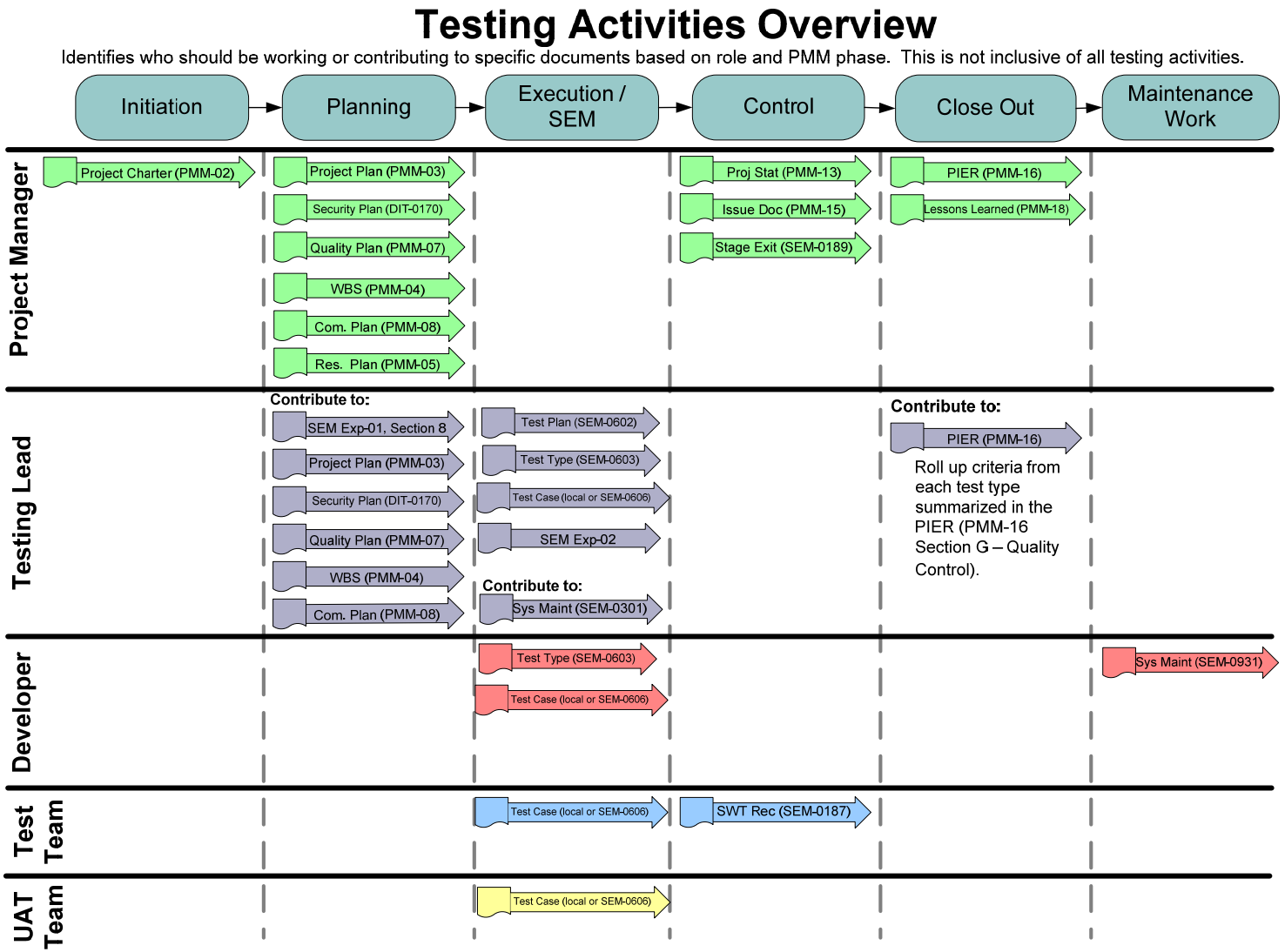
Tasks: Ensure that:

- The Test Plan, all Test Type Approach and Report and Test Case documents are listed as tasks.
- Time required to plan testing activities is included as a task.
- All activities for each test type are listed as tasks.
- Data Conversion tasks are included and properly sequenced with testing activities, if applicable.
- There are Structured Walkthrough tasks for each specific test type.
- The Stage Exit meeting is included as a task.

Included on the next page is a PMM visual representation of what documents are completed when and by whom.

Notes: The Phases (columns) are PMM Phases not SEM Stages.

Exhibit 2.0-1 SEM Testing Activities Overview Diagram



Chapter: **3.0 Requirements Definition Stage**

Key Documents: Requirements Definition Stage documents related to testing:

- Requirements Traceability Matrix (SEM-0401)
- Requirements Specifications (SEM-0402)
- Project Lessons Learned (PMM-18)

Document: **Requirements Traceability Matrix (SEM-0401)**

Tasks: Map the requirements back to the system/project objectives identified in the Project Plan. During test case creation, ensure there are test cases to test each requirement, as prioritized by the client.

Notes: Not all requirements have an associated test case. Timeframe crunches or other factors may limit the number of requirements that are testable. When this situation arises, the project group must agree on a prioritized subset of requirements to test (e.g., high – medium risk or level of importance).

It is important to document the agreed-upon decision in Section 1.3 Out of Scope of the Test Plan. Enter a comment in the Requirement Modification(s)/Comments column within Section 3. Requirements Traceability Matrix of the Requirements Traceability Matrix for each requirement excluded from testing.

Document: **Requirements Specifications (SEM-0402)**

Tasks: Ensure each requirement is "testable". As requirements are identified, always ask: "How are we going to test for this requirement?"

Notes: All requirements may not be testable. If a non-testable requirement is going to be retained, a testing mitigation strategy should be noted under the Requirement Modification(s)/Comments column within Section 3. Requirements Traceability Matrix of the Requirements Traceability Matrix.

Examples: Due to a project time crunch, the testing schedule needs to be condensed. As a result, testing of custom yearly reports cannot be completed prior to system implementation. The mitigation strategy to this situation is to perform post-implementation testing of these reports because the business side will not need the custom yearly reports for six months.

Document: **Project Lessons Learned (PMM-18)**

Tasks: Document lessons learned collected from participants throughout the project.

Notes: Incremental lessons learned meetings or surveys should be completed throughout the development lifecycle. It is best to gather incremental lessons learned after each phase, instead of waiting until the end of the project.

Chapter: **4.0 Functional Design Stage**

Key Documents: Functional Design Stage documents related to testing:

- Functional Design Document (SEM-0501)

Document: **Functional Design Document (SEM-0501)**

Tasks: Reference the information entered in the Functional Design Document (or Use Cases, if used) as input to creating test case(s).

Reference the System Design document as input to creating test case(s).

Review the Requirements Traceability Matrix to ensure specifications indicate how the functional design will meet the requirements.

Notes: Do all that is possible to prevent scope creep.

Chapter: **5.0 System Design Stage**

Key Documents: System Design Stage documents related to testing:

- System Design Document (SEM-0604)
- Conversion Plan (SEM-0601), if applicable
- Test Plan (SEM-0602)
- Test Case (SEM-0606 or equivalent), multiple
- Test Type Approach and Report (SEM-0603), multiple
- Software Configuration Management Plan (SEM-0302)

Document: **System Design Document (SEM-0604)**

Tasks: Review Modules and Processing Narrative for the design flow which can be used as input into Section 2.2 Modules to be Tested of the Test Plan (SEM-0602).

Document: **Conversion Plan (SEM-0601)**

Tasks: Review Sections 3.3 Output Data and 3.4 Validation as input to creating test case(s).

Document: **Test Plan (SEM-0602)**

Description: The Test Plan is created to communicate the overall testing process to all appropriate stakeholders (e.g., project team, business owners, sponsors, management). It provides a road map to:

- Who participates
- What to test
- When and where testing will take place
- How testing will be conducted, tracked and recorded

Tasks: Check the types of testing to be conducted for the project in Section 1.1 Test Types to be Performed.

Ensure that:

- Section 1.2 Testing Scope describes, at a high level, what testing is to be completed. This should be used as input into Section 1.2 Purpose and Scope of the Test Type Approach and Report document (SEM-0603).
- Section 1.3 Outside of Scope identifies any items that will not be tested.
- Section 1.4 Testing Resources lists the personnel, their roles and applicable test type(s) each resource will perform. Reference Section B. Resource Profiles in the Resource Plan (PMM-05) for initial testing contact(s).

- Section 1.4.1 Training indicates how the testers will be trained on the test processes, testing tools, the application being tested, and defect reporting.
- Section 2.1 Test Environment(s) describes servers, databases and tools that will be needed for the testing effort. Ensure the release schedule to the test environment(s) is clearly defined and agreed upon between developers and testers.
- Section 2.2 Modules to be Tested identifies the high level functionalities that need to be tested. List them in the order that testing should occur. Reference the System Design Document (SEM-0604) to ensure coverage.
- Section 3.1 Test Schedule includes test resources, start/completion dates for each test activity, test deliverable(s), and milestone(s). Include reviews, agreements to all testing approaches, and Stage Exit(s).
- The Project Plan (PMM-03) is updated with all testing activities (e.g., test resources, start/completion dates for each test activity, test deliverable(s), and milestone(s)).
- Section 3.2 Monitoring describes methods that will be used to evaluate test progress against the project plan.
- Section 3.3 Defect Reporting describes the defect reporting process, including how defects will be reported, as well as where and how the defects will be updated and stored.
    - Document and agree upon the descriptions of the values (e.g., severity, priority), used in your defect reporting process.

Check off all documents that will be used throughout testing in Section 4. Testing Documentation.

Document: **Test Case (SEM-0606 or equivalent), multiple**

Description: Each test type has its own set of test cases. Each set of test cases should be written at a level of detail appropriate for the group conducting the testing. This concept relates back to the V-Model diagram in Section 1.5 Testing Methodologies of this document. The developers will have a set for their unit, functional and integration testing that should be very detailed (white-box testing). The QA testers will have a set for system and standards testing that should exercise the system from end-to-end (black-box testing). The end-users will have a more test-scenario based set for User Acceptance Testing (UAT). This will ensure that all requirements are met in a manner that meets their business needs.

For each set of test cases, inputs will come from the Requirements Specification (UAT), Functional Design (Functional and System Test) and System Design (Unit and Integration Test) documents.

There are many ways of documenting the various test types. The important thing is to document them. The Test Case document is a generic template that can be used or referenced. It highlights the items that should be captured for the different types of test cases.

Tasks: Create test cases that cover all requirements.

Document: **Test Type Approach and Report (TTAR) (SEM-0603), multiple**

Description: The intent of this document is to have the agreed-upon testing scope, approach, success criteria, expected results, actual results and documentation to support the final decision to proceed to the next test type or implementation stage in a single location.

It is at the discretion of the test lead and project team to decide how many different TTAR documents should exist for a given project. At a minimum, the test types selected in Section 1.1 Test Types to be Performed in the Test Plan (SEM-0602) must be covered in a TTAR. However, due to an application's complexity, there may be different TTARs broken out by the specific module(s), component(s), functionality, etc.

Examples: On a straightforward, low-complexity project, you may have TTARs that cover the following test types:

- Unit
- Integration
- Functional
- System and Standards
- Performance
- User Acceptance

A more complex project may require multiple sets of TTARs, one set for each specific module, component, functionality, etc. For a project with two distinct functionalities, the following might apply for each test type:

- 2 TTARs – Unit
- 2 TTARs – Integration
- 2 TTARs – Functional
- 2 TTARs – System and Standards
- 2 TTARs – Performance
- 2 TTARs – User Acceptance

There could be more then one set of TTARs, if the testing will be performed in iterations. Each iteration should be documented in a separate TTAR.

Another scenario where a new TTAR would be created is if, at the end of a test type, the decision was a "No-Go" (to not move forward). Initially, document the steps needed to reach a "Go" decision in the current TTAR under review. Then transfer them into Section 1.2 Purpose and Scope of the new TTAR. This way, it

keeps a history of the testing activities, as well as expected and actual results throughout the project.

Tasks: Quantify the Purpose and Scope to ensure it matches the indicated test type.

Identify all testing activities (e.g., tester training, environment setup, data setup, documentation setup, execution and reporting) in Section 2.1 Test Approach.

Identify criteria for successful completion (Section 2.2 Success Criteria).

Indicate who will participate in the testing and identify their corresponding roles (Section 2.3 Test Participants).

Calculate the expected results (Section 2.4 Expected Results) using the following formula: Defined Scope (Section 1.2 Purpose and Scope) multiplied by the Success Criteria (Section 2.2 Success Criteria) equals the expected results.

Document the decision regarding the Test Approach (Section 2.5 Test Approach Decision).

Document: **Software Configuration Management Plan (SEM-0302)**

Tasks: Ensure testing stakeholders are trained and knowledgeable on the team's Software Configuration Management (SCM) processes and tools.

Identify testing documentation as Configurable Items (CIs) in Section 3.2 Configuration Items (CIs).

Ensure a role for the testing stakeholder is included on the Local Change Board (Section 2.1.4 Local Change Board (LCB)).

Chapter: **6.0 Construction Stage**

Key Documents: Construction Stage documents related to testing:

- Test Plan (SEM-0602)
- Test Type Approach and Report (SEM-0603), multiple
- Test Case (SEM-0606 or equivalent), multiple

Document: **Test Plan (SEM-0602)**

Description: See Chapter 5.0 System Design Stage, Test Plan Document Section for the document description.

Tasks: Finalize Section 1.4 Testing Resources.

Deliver test team training on test and release processes, testing tools, servers, databases, the application being tested, and defect reporting.

Determine the order of the testing modules.

Update the Project Plan (PMM-03) with all testing activities (e.g., test resources, start/completion dates for each test activity, test deliverables, and milestones).

Review Test Environment to ensure validity.

Review Section 2.2 Modules to be Tested to ensure validity and that no unauthorized changes have been made that are not in the system design specifications.

Define how the methods will be used to evaluate test progress against the Project Plan in Section 3.2 Monitoring.

Place the Test Plan (SEM-0602) under configuration management when approved after the Stage Exit.

Document: **Test Type Approach and Report (TTAR) (SEM-0603), multiple**

Description: See Chapter 5.0 System Design Stage, Test Type Approach and Report Document Section for the document description.

Tasks: Ensure one TTAR (SEM-0603) document, at a minimum, covers each test type checked in the Test Plan (SEM-0602).

Verify that Section 1.1 Test Type identifies the particular test type to be executed.

Ensure that:

- Section 1.2 Purpose and Scope is quantifiable for the test type identified in Section 1.1 Test Type.
- Section 2.1 Test Approach explains the steps needed to complete the test type (e.g., tester training, environment setup, data setup, documentation setup, execution and reporting).
- The steps identified in Section 2.1 Testing Approach are Specific, Measurable, Achievable, Realistic and Time-based (SMART). No steps should be unclear or high-level.
- Section 2.2 Success Criteria describes test conditions needed to advance the project.
- Section 2.3 Test Participants identifies team members and roles by name.

Review the calculation of the expected results to ensure it is still applicable.

Ensure acceptance of the test approach prior to testing execution.

Place the TTARs (SEM-0603) under configuration management once the approach sections have been approved.

Document: **Test Case (SEM-0606 or equivalent), multiple**

Description: See Chapter 5.0 System Design Stage, Test Case Document Section for the document description.

Tasks: Create the test cases with the following information:

- A high-level description of the test case goal.
- Any setup requirements needed to run the test.
- Security role of the tester.
- Steps needed to complete the test (detailed or general depending on test type).
- Expected outcomes.
- Pass/Fail history and test date.

Chapter: **7.0 Testing Stage**

Key Documents: Testing Stage documents related to testing:

- Test Type Approach and Report (SEM-0603), multiple
- Test Case (SEM-0606 or equivalent), multiple
- Requirements Traceability Matrix (SEM-0401)
- Defect Tracking Log (SEM-0186 or equivalent)
- Structured Walkthrough (SEM-0187), multiple

Document: **Test Type Approach and Report (TTAR) (SEM-0603), multiple**

Description: See Chapter 5.0 System Design Stage, Test Type Approach and Report Document Section for the document description.

Tasks: Summarize the actual results and the success rate (Section 3.2 Results Summary), as measured against the scope, using a defect tracking tool (SEM-0186 or equivalent).

Analyze the outstanding defects, if applicable, and utilize this information as input into the Go/No-Go decision to proceed.

Document the Go or No-Go decision in Section 3.2.2 Go/No-Go Decision.

Obtain approval (Approval Information Section) for the agreed-upon decision.

Document: **Test Case (SEM-0606 or equivalent), multiple**

Description: See Chapter 5.0 System Design Stage, Test Case Document Section for the document description.

Tasks: Execute the test cases.

Record test outcomes.

Report data to the resource who monitors progress.

Record defects in the agreed-upon defect tracking tool.

Document: **Requirements Traceability Matrix (SEM-0401)**

Description: The Traceability Matrix is to be used to map the requirements back to the system/project objectives identified in the Project Plan. Update this document (with the requirement's status, based on test outcomes) throughout the testing lifecycle, as you progress from one test type to another. In preparation for each test type's Go/No-Go decision, ensure requirements are documented in the

Traceability Matrix (SEM-0401). This document should be under version management and checked in and out to keep it current.

Tasks: Update Test Reference Number(s) in the table.

Update Status in the table.

Document: **Defect Tracking Log (SEM-0186 or equivalent)**

Description: The objective of recording defects is to produce a complete record of the differences between the actual and expected results for each test case. The Defect Tracking Log is used to report the discrepancies found.

Reasons for recording defects include: correction of the defect, reporting on the application status, and gathering metrics to assist in test planning for future applications.

The tools used to consolidate and record defects depend on your local agency.

Tasks: Create or update the Defect Tracking Log.

Document: **Structured Walkthrough (SEM-0187), multiple**

Description: As each test type is completed, an informal Structured Walkthrough meeting must be held with key stakeholders to review the TTAR document (SEM-0603). The goal is to review the results and reach a Go or No-Go decision.

If a "Go" decision is reached, the next test type can begin. When the decision is made regarding the final test type, the project can proceed to the implementation stage.

If a "No-Go" decision is reached, the next steps need to be agreed upon, in order to move toward reaching a "Go" decision. A new TTAR document (SEM-0603) must be created to capture the agreed-upon next steps, expected results and actual results.

Tasks: Complete Section 3. Results Report of the TTAR document (SEM-603).

Attend/Conduct Structured Walkthrough Meeting.

Chapter: **8.0 Implementation Stage**

Key Documents: Implementation Stage documents related to testing and quality:

- Project Lessons Learned (PMM-18)
- Post Implementation Evaluation Report (PMM-16)

Document: **Project Lessons Learned (PMM-18)**

Tasks: Document lessons learned collected from participants throughout the project

Notes: Incremental lessons learned meetings or surveys should be completed throughout the development lifecycle. It is best to record incremental lessons learned after each phase, instead of waiting until the end of the project.

At the time of project close, review, evaluate and adjust the lessons learned, as needed.

Examples: A lesson learned that was identified at the end of the System Design Stage might be: "Having to complete the Test Plan in System Design seemed premature." However, by the end of the project, the team realized the value of completing the Test Plan in that Stage because it enabled them to line up the correct testing resources and flush out testing roadblocks."

Document: **Post Implementation Evaluation Report (PMM-16)**

Tasks: Evaluate the items in the Project Lessons Learned document (PMM-18) to identify which ones provided the most value. Communicate these findings to the project stakeholders.

Record these findings in the Post Implementation Evaluation Report (PMM-16) under Section K. Lessons Learned.

Appendix A – Key Terms and Acronyms

Agile Methodology
A group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.

Automated Testing
The use of software to control the execution of tests, comparison of actual outcomes to predicted outcomes, setting up test preconditions, and other test control and test reporting functions.

Black-Box Testing
Black-box testing implies that a tester does not know how an application is designed at the code level. The tester interacts with the software system via its interface and analyzes the application reaction. The tester uses specific input in an attempt to get expected output.

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

Defect
A lack of something necessary for completeness, adequacy, or perfection (a deficiency).

Term used to describe an error, flaw, mistake, failure, or fault in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways. Also see Defect Reporting.

Defect Reporting
Recording the defects, identified at each stage of the test process, to create a complete record of the discrepancies identified during the life of the project. This is especially critical during the testing stage. The information captured identifies who reported the defect, when it was discovered, a description of the defect and the steps required to reproduce the problem, if applicable. The rest of the information captured is the status, priority and severity of the defect, who the defect was assigned to, when it was fixed and a description of the resolution.

Development Lifecycle
A process of creating or altering systems, which includes the models and methodologies that people use to develop these systems.

Expected Outcome
The behavior predicted by the specification of a product under specified conditions (predicted outcome).

Expected Results
Defines a tolerable success rate that will render the system acceptable and operational. Expected results are the threshold of acceptance used to measure against actual results.

Function Test
Confirms that the logically-grouped modules function according to specifications. Developers write this test from a user's perspective. Testing is based on output only, without any knowledge of internal code or logic. Function tests tell a developer that the code is *working properly*.

Go/No-Go Decision
Determination to proceed with or abandon a plan or project. In Quality Control, "Go" denotes that a product conforms to the specifications or the agreed-upon success criteria has been met. When it does not, it is "No-Go".

Integration Test
Verifies that the system components are integrated and working as an application. The technical development team performs this test to uncover errors that occur in the interactions and interfaces between components.

Iteration
The process of repeating a set of instructions either a specified number of times or until a specific result is achieved.

Iterative Development
An approach to application development in which prototypes are continually refined into an increasingly complete and correct system.

Module
A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading. A logically separable part of a program.

Performance Test
Measures software performance of batch data, under actual or anticipated volume, as well as on-line transaction response times. Executed by the technical team, this test verifies performance requirements, throughput and growth capacity. Performance tuning continues throughout the system lifecycle.

Project Management Methodology (PMM)
A component of the of the State Unified Information Technology Environment (SUITE) which provides standard methods and guidelines to ensure that projects are conducted in a disciplined, well-managed, and consistent manner that promotes the delivery of quality products that meet the customer's needs and results in projects that are completed on time and within budget.

Priority (defects)
A preferential rating; the ranking that is given to a defect to signify the level of importance (i.e., Low, Medium, High).

QA Build
When the executable code (complied code) is deployed to the QA/Test environment.

QA Build Cycle
The frequency with which the code is promoted (e.g., daily or weekly) to the QA/Test environment.

Quality Assurance (QA)
A process designed to provide management with appropriate visibility into the work products being built and the systems engineering processes being used by the project team. It is one of the Software Engineering Institute Capability Maturity Model Integrated (CMMI) Level 2 key process areas.

Quality Control (QC)
Processes to find and fix defects (e.g., walkthroughs and testing).

Quality Improvement (QI)
A process to reduce variation and defects in any process or product.

Regression Test
Re-execution of specific test cases to ensure defects are fixed, find new defects that may have been introduced, and confirm that module(s) are functioning properly. Any testing stakeholder can conduct this test.

Severity (defects)
Of a great degree; a classification of the degree of impact that error or fault has on the product (i.e., Low, Medium, High and Show-Stopper).

Specific, Measurable, Achievable, Realistic and Time-based (SMART)
Standard criteria to review requirements against.

State Unified Information Technology Environment (SUITE)
The State of Michigan's implementation of a standard methodology, procedures, training, and tools for projects 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 the Capability Maturity Model Integrated (CMMI) Level 3 requirements.

Structured Walkthrough (SWT)
An organized procedure for a group of peers to review and discuss the technical aspects of software development work products. The major objectives of a structured walkthrough are to find errors and to improve the quality of the product.

Success Criteria
The metrics and measurements established to determine whether the product has satisfied its objectives. It is also used as input to determine if the project should proceed to the next step.

System & Standards Test
Verifies functional business requirements, business processes, data flows and other system criteria are met. The technical team tests specific end-to-end business processes until the complete application environment mimics real-world use. Verifies that Federal, State of Michigan and department standards are met.

Systems Engineering Methodology (SEM)
A component of the State Unified Information Technology Environment (SUITE) which provides guidance for information systems engineering related project management activities and quality assurance practices and procedures.

Test Case
A set of conditions or variables under which a tester determines whether or not an application or software system is working correctly. It is the mechanism for determining whether a software program or system has passed or failed. It can require many test cases to determine that a software program or system is functioning correctly.

Test cases are also often referred to as "test scripts", particularly when written. Another common term is "test scenarios", which is often the term used when referencing User Acceptance Testing (UAT).

Test Plan
The strategy used to verify and ensure that a product or system meets its design specifications and other requirements.

Test Script or Test Scenario
See Test Cases.

Test Type Approach and Report (TTAR)
A document which contains an agreed-upon testing scope, approach, success criteria, expected results, actual results and documentation to support the final decision to proceed to the next test type or implementation stage.

Testing Lifecycle
A testing process which is defined and has structure.

Unit Test
Confirms that the program logic within an application module produces the expected output when given a known input. Written from a developer's perspective, this ensures that the smallest testable module of an application successfully performs a specific task(s). Unit tests tell a developer that the code is *working properly*.

Use Case
A description of a system's behavior as it responds to a request originating from outside of the system (e.g., a user). In other words, a use case describes "who" can do "what" with the given system.

User Acceptance Test (UAT)
Validates the system as a whole meets mutually agreed-upon requirements. UAT is completed by end-users of the application and occurs before a client or customer accepts the new system.

User Stories
A software system requirement formulated as one or more sentences in the everyday or business language of the user.

V-Model (software development)
A software development process which can be presumed to be the extension of the waterfall model. Instead of moving down in a linear fashion, the process steps are bent upwards after coding to form the typical V shape. The V-Model demonstrates the relationships between each stage of the development lifecycle and the associated stage of testing.

The V-model deploys a well-structured method in which each stage can be implemented by the detailed documentation of the previous stage. Testing activities such as test designing are initiated at the beginning of the project well before coding and they can therefore save a huge amount of project time.

Validation & Verification
The process of checking that a product, service, or system meets specifications and that it fulfills its intended purpose. These are critical components of a quality management system such as ISO 9000.

Verification Test
Verifies that a product or product component fulfills its intended use when placed in its intended environment.

Waterfall Development Methodology
A sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design (validation), Construction, Testing and Maintenance.

White-Box Testing
In this case a tester knows the internal program structure and its code. As a result, the tester can execute each program statement and function, check intended error handling, etc. This testing involves source code reviews, walkthroughs, as well as design and execution of tests, based on the access to the program code.

Appendix B – Bibliography

The following materials were referenced in the preparation of this process manual.

Carnegie Mellon University, Software Engineering Institute (SEI) website: http://www.sei.cmu.edu/index.html

Chrissis, Mary Beth; Konrad, Mike, Shrum, Sandy (2007). *CMMI Second Edition Guidelines for Process Integration and Product Improvement 1.2*, Upper Saddle River, NJ: Addison-Wesley.

Black, Rex, (2004). *Critical Testing Processes; Plan, Prepare, Perform, Perfect,* Boston: Addison-Wesley.

Stanton-Reinstein, Dr. Rebecca (2006). *Software Quality Assurance Methods and Techniques Training provided through International Institute for Software Testing.*

From V to I: How Agile Practices Reduce Cycle Times. Toolbox for IT website: http://it.toolbox.com/

V-Model (software development) 2009, September). In Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/V-Model_(software development)