Chapter 12

Testing

CONTENTS

This page intentionally left blank.

Chapter 12

Testing

"We test because we know that we are fallible …" – Paul C. Jorgensen

12.1 Introduction

Testing is not new to any of us. It comes almost daily and can be formal or informal. Our steps through school and training are marked by tests. For some, each day brings a test of survival. Progress is never realized without some form of testing. This maxim also holds true for software development projects.

While failure to properly implement a functional test program for software used by a commercial enterprise can lead to loss of revenue and the demise of the company, failure in a military environment can lead to mission failure, precipitate injury and death, and ultimately jeopardize the survival of the nation. This chapter discusses testing principles and processes, particularly as they relate to software testing. However, the concepts can also be readily applied to other types of systems. Because testing is part of a comprehensive assessment and evaluation program, you are encouraged to also read Chapter 11, "Assessing Project Health," if you have not already done so.

A good testing program is essential for reliable operational performance, and will significantly reduce support and maintenance costs. Properly implemented testing significantly improves the probability of project success, thereby enhancing the likelihood of mission success. Test planning and preparation impact the project early by helping define requirements that are testable. As products are being developed, testing allows the developers to discover and fix problems before they can become showstoppers. While a good testing program costs money and can't necessarily put a troubled project on the road to recovery by itself, it does provide extra insurance that most project managers need by helping to signal and avoid trouble. [1]

Testing has two fundamental purposes: [2]

1. To evaluate quality or acceptability of that which is being tested.
2. To discover problems or errors.

These primary purposes can be further expanded into three groups of objectives, shown in Table 12-1.

Table 12-1 Objectives of Testing [1]

Demonstration	Detection	Prevention
• Show that the system can be used with acceptable risk. • Demonstrate functions under special conditions. • Show that products are ready for integration or use.	• Discover defects, errors, and deficiencies. • Determine system capabilities and limitations. • Determine quality of components, work products, and the system.	• Provide information to prevent or reduce the number of errors. • Reduce the number of early errors propagated through to later phases. • Clarify system specifications and performance. • Identify ways to avoid risks and problems in the future.

12.2 Process Description

Testing is one of the major inputs to project health, serving not only as an indicator of, but also as a primary contributor to, the health of the project by improving quality and discovering errors and risks before they become critical problems. Because a testing program is a major part of the project it should be well defined and documented. It should be considered in and influence every major phase of the project, as shown in Figure 12-1.

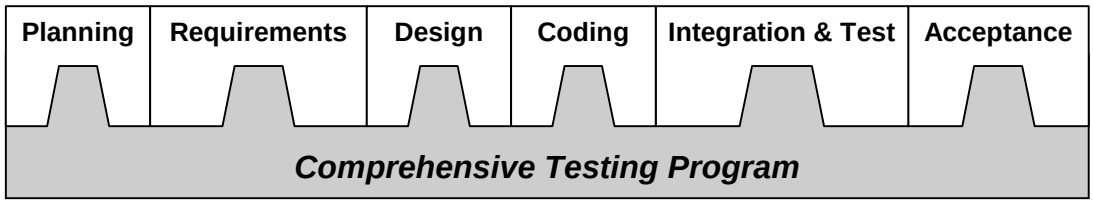

Figure 12-1 An Effective Testing Program is Part of all Project Phases

Testing is accomplished by implementing a testing program. A good program includes the right people, methods, tools, and equipment needed to perform the testing, along with a testing process to integrate all these into an efficient, effective program, as shown in Figure 12-2. [3] Testers should be trained, skilled, and motivated people. Employing people who lack these attributes will increase the amount of time and resources needed for testing and increase the risk of overlooking important testing considerations.

Because testing is an integral part of the development process, adequate time and resources are reserved for testing. Tests also serve as milestone and progress indicators. Proven methods should be used to facilitate efficient, effective testing. It must be remembered that everything cannot be tested. Even if it were possible to think of all possible test cases, the effort would consume over 90% of a project's resources and take an unacceptable length of time. Because of this, testing must be risk-based, where those areas with the greatest risk of failure are given the most attention. Effective testing is creative and uses careful analysis, planning, and design to select test strategies, methods, and tests to get the greatest amount of evaluation for the testing resources available. [1]

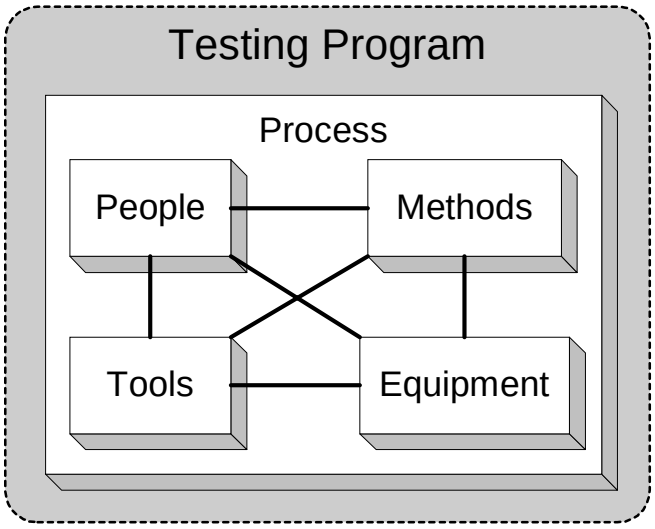

Figure 12-2 Elements of an Effective Testing Program

Appropriate equipment should be available and properly used to allow thorough testing. Without the right equipment, testers may have to use slower, roundabout testing methods, taking longer to test, perhaps missing important test results, or even bypassing important tests.

Test tools cover a broad spectrum of testing assistants. They range from simple databases to track tests, results, and requirements met, to full automatic testing systems that perform comprehensive testing, including varying the logical flow of testing, depending on test results. Tools may be hardware based, software based, or both. The choice of tools requires consideration be given to the type of testing, the amount of testing, testing expertise of the organization, costs, schedule, possibility for reuse, and requirements. A commonly used tool is a tracking system for recording what has been tested and what has not. Additionally, a database of test results is maintained for current and future reference. Test results are used as a feedback mechanism not only to determine which components need to be reworked, but for improving the development process and methods in the current and future projects.

Too often, testing is viewed as a hurdle to get over, rather than a tool for improving both product quality and the development process. When testing is considered and planned early in the project, along with establishing require-

ments that are testable, it can provide guidance throughout the project and reduce the time required for rework. Early life cycle testing helps prevent propagation of defects to later stages of development.

12.2.1 Establishing a Test Process

The testing process is the keystone to effective testing. It defines how test components are used with each other, and what is to be done at what times. The test process should be active throughout the entire development life cycle as indicated by Figure 12-1. It begins with planning and continues on through the acceptance of the developed product. Table 12-2 lists the major testing activities performed during each major phase of the project.

Table 12-2 Testing Activities of Project Phases

Phase	Testing Process Activity
Planning	Testing activities are scheduled throughout the project. Testing to meet requirements is established as a project priority.
Requirements	Requirements are written so that they are testable.
Design	System and components are designed to be testable and to meet requirements by passing tests.
Coding	Testing is performed in the code-test cycle (see Figure 12-6) and on complete software units. Debugging and regression testing is also performed.
Integration & Test	Testing is performed on integrated subsystems and systems (integration and system testing.) Debugging and regression testing are also performed. (See Figures 12-3 and 12-6.)
Acceptance	User acceptance testing is performed.

12.2.1.1 Testing Levels

A testing process should also follow a hierarchy of testing (shown in Figure 12-3) where components are tested at the lowest level and then at successively higher levels of integration until the complete system is tested and debugged. Finally acceptance testing is performed.

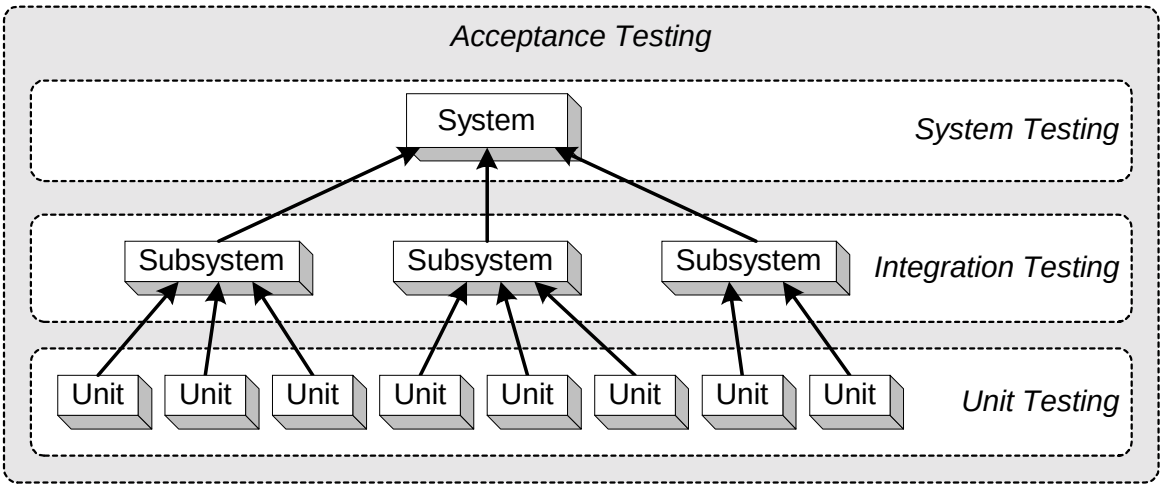

Figure 12-3 Testing Hierarchy

12.2.1.2 The Testing Process

The overall test process is shown in simplified form in Figure 12-4. Because it has already been discussed, the process leaves out the consideration of testing in establishing requirements and in design. After completion of the first (planning) activity, all other activities are repeated for unit, integration, and system testing. Because of the code-test-debug cycle (shown in Figure 12-6), there may be several iterations of coding and integration, along with their accompanying testing.

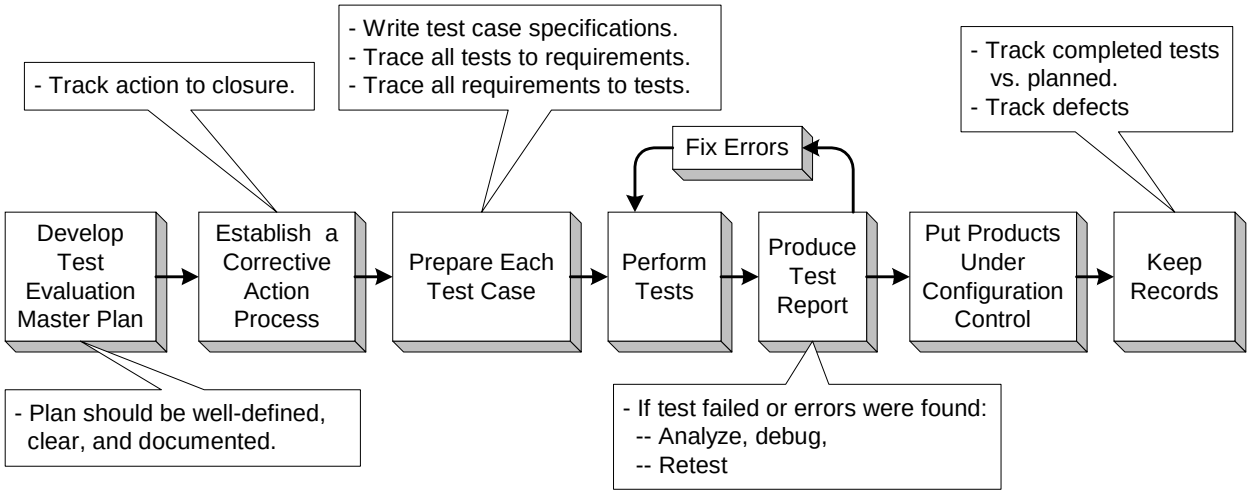

Figure 12-4 Simplified Testing Process

12.2.1.3 Test Planning

When planning the testing program, a hierarchy of plans is involved in the process, each providing test direction at different levels of the system and detail. Figure 12-5 depicts the Test Planning Tree as defined by the Software Program Managers Network (SPMN). The contents of these planning documents which relate to testing are summarized in Table 12-3. [4] More can be found on this and other testing principles in the SPMN's highly recommended *Little Book of Testing*, Volumes I and II. [1]

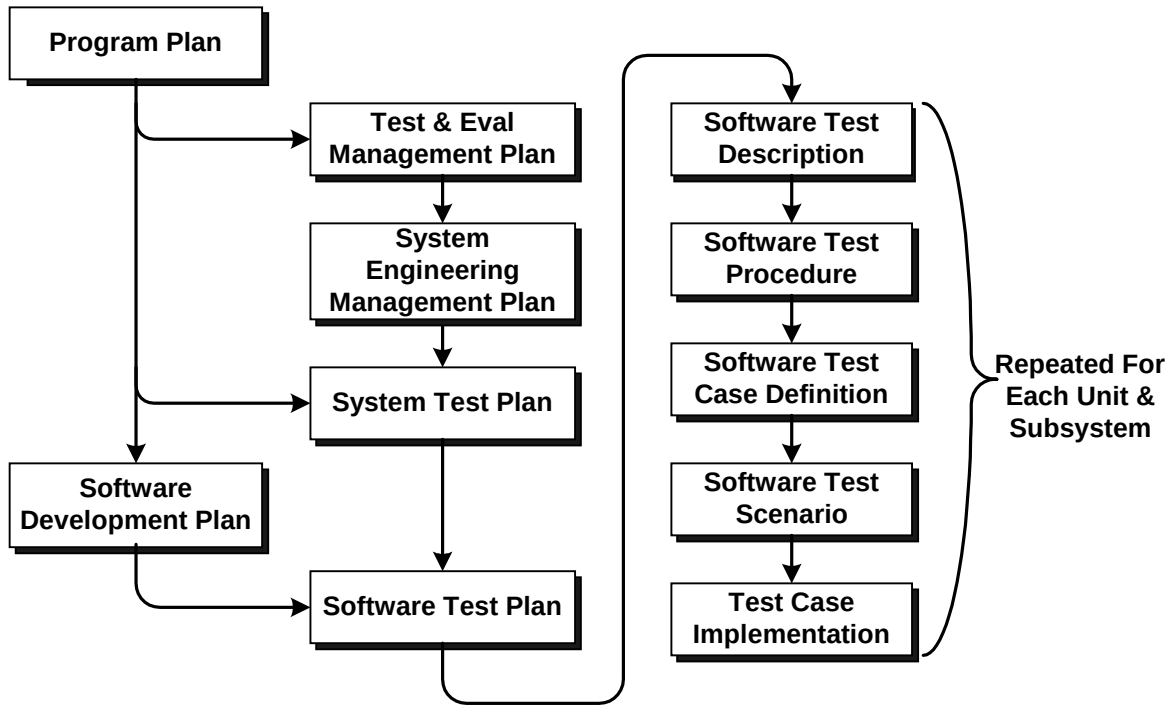

Figure 12-5 Test Planning Tree [4]

Several of these documents may be combined where appropriate. Often, the test procedure will contain the information for the case definition, test scenario, and case implementation.

Table 12-3 Testing Related Content of Project Planning Documents [4]

Planning Document	Testing Related Content
Program Plan	Defines program requirements, strategies, etc. Identifies test requirements.
Test & Eval Management Plan	Establishes entire testing strategy and practices.
System Engineering Management Plan	Identifies plans, methods, standards, processes, and systems for integration and test requirements.
System Test Plan	Defines system testing strategies, controls, and processes. Test case requirements are discussed and success criteria are identified.
Software Development Plan	Describes the criteria for overall integration, test, and completion.
Software Test Plan	Defines software and integration test plans, strategies, controls, and processes. Also defines test case requirements and completion criteria.
Software Test Description	Describes the requirements for individual test cases.
Software Test Procedure	Provides a step-by-step procedure for executing the test. It traces to a requirement specification.
Software Test Case Definition	Defines the test case and specifies success criteria for a test step.
Software Test Scenario	Identifies specific data, sources, interfaces, timing, expected response, and the next step to take in response to a test condition.
Test Case Implementation	Executable test case.

12.2.2 Types of Testing

Many different types of testing are used to fulfill the many different testing needs. Several of the major types are summarized here. Figure 12-6 indicates the phase of the project where some of these are generally found. Note that test types may be found in more than one phase. Remember that maintenance projects are usually considered as development projects also, ending with special maintenance and regression testing.

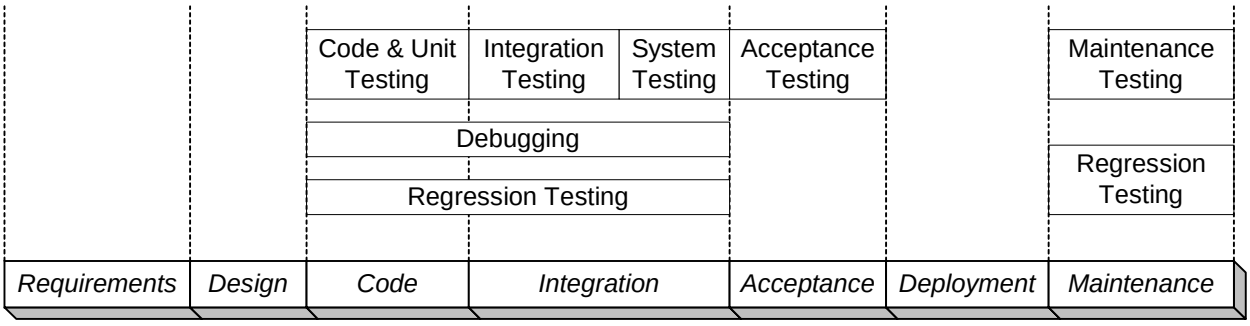

Figure 12-6 Types of Testing Performed During Different Project Phases

12.2.2.1 Debugging

Debugging is testing used from the unit to the system level to determine what is causing errors. It consists of search methods used to isolate problems to a specific module or cause. When the problem is found it is fixed and retested. While a formal test failure is usually an indicator of errors, debugging often involves a great deal of "free-form" testing by software and systems engineers. While training can help, most expertise in debugging comes from experience.

12.2.2.2 Code and Unit Testing

As a software engineer codes a software module or unit, there will usually be testing of different pieces along the way to make sure they are accomplishing the purpose of the unit. This is code testing and will consist mostly of informal, free-form testing. Unit testing is performed when the unit is believed to be complete. It is tested to make sure all inputs are handled correctly, producing the correct outputs with the proper timing, etc. Unit testing is usually more formal because the unit will probably need to handle specific inputs and produce specific outputs or actions. When units are complete, they are ready for the integration process.

It should be noted that unit testing is not always as straightforward as the name implies. Each unit should be well defined before the testing is defined. Not every module or source file is necessarily a unit, especially when considering code maintenance or enhancement. A unit may consist of a single module, part of one module, or may be comprised of several modules and their associated files. A unit may overlap several requirements, or a single requirement may involve several units. A system architect may be necessary to assist the test definition process.

12.2.2.3 Integration Testing

While software modules may function well by themselves when they are developed, getting them to work together efficiently and correctly is another matter. After they have been coded and tested individually, individual software components are combined to form a final software product. During this integration effort, tests are performed on various groupings of components to determine how well they work together. Incompatibilities, errors, and inadequacies are discovered and fixed. Eventually, all software modules are integrated and debugged so they function correctly as a whole.

12.2.2.4 System Testing

When the software, hardware, and other subsystems are complete, they in turn are integrated and tested as a system. This is the final development testing. Any problems or errors discovered during systems testing are analyzed to determine which subsystems are at fault, then those subsystems are sent back for debugging, with its attendant code, unit, and integration testing. Figure 12-7 shows the various levels of testing associated with development and how problems and errors feed back to earlier developmental stages. System testing evaluates the functionality of the system, including capabilities, compatibility, stability, performance, security, and reliability. [5]

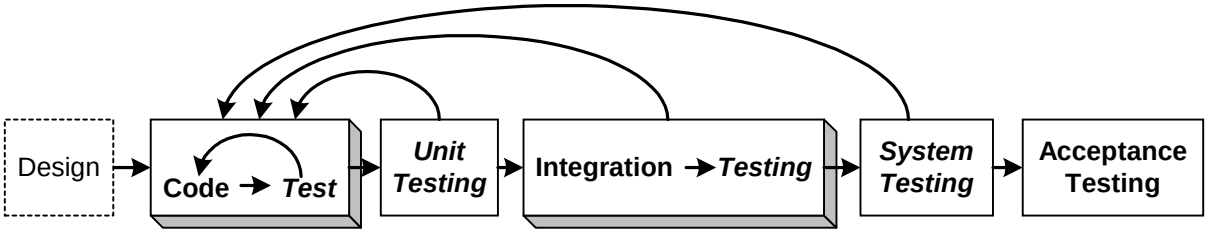

Figure 12-7 Development, Test, and Debug Cycle

12.2.2.5 Acceptance Testing

Acceptance testing is that formal demonstration that the system performs according to requirements. It should be rehearsed during the system testing phase so that there are no failures or problems to be discovered. Acceptance testing is usually witnessed by the customer and other stakeholders. A failure at this point is usually indicative of incomplete testing in the development phase. Because the format and procedures of the acceptance tests have been coordinated and agreed upon among developers and stakeholders, successful completion of the test should signal acceptance by the customer and clear the way for deployment.

12.2.2.6 Regression Testing

Regression testing is not associated with any particular stage of the project, but should be performed whenever there has been a change to a component or the system. It consists of testing components or a system after changes have been made to verify that the components or systems still comply with the requirements and that the modifications have not caused unintended effects. [5]

12.2.2.7 Maintenance Testing

Maintenance testing must be performed anytime there is a maintenance upgrade to the system. Its purpose is to ensure the new modifications are properly integrated and work with the rest of the system. It also verifies that the upgrade provides the additional functionality the maintenance upgrade is supposed to add. Maintenance testing should include regression testing to ensure the upgrade does not cause any undesired effects. [5]

12.2.2.8 Alpha Testing

Alpha testing is a preliminary field test by a select group of users with the purpose of finding bugs not discovered through previous testing and to refine the user interface. It is an extension of system testing and may or may not be used, depending on the project and product. The product is complete by this time but not necessarily refined. The test group is usually made up of people within the developing organization, but not the developers themselves. [5]

12.2.2.9 Beta Testing

Beta testing is similar to and performed following alpha testing. Like alpha testing, it is optional, depending on the project. The key difference is that the testers consist of selected users outside the developer's organization. [5]

12.2.2.10 Black box testing

Black box testing is testing the function of a component or system from a user's point of view without regard to the internal structure or logic involved. [5] This should be done at various times throughout the development to maintain an understanding of the user's perspective and meet the user's needs as well as the requirements.

12.2.2.11 Stress Testing

Stress testing is a form of system testing. The systems or subsystem is tested under extreme or abnormal conditions outside the operational envelope with the purpose of finding the limits where the item being tested fails or breaks down. This enables the testers to determine how much margin there is between expected operating conditions and failure conditions. Stress testing may also be used to determine sensitivity, which types of conditions or combinations thereof affect the system most and least. [6]

Stress testing of hardware may include vibration, pressure, temperature, and other environmental conditions. In software systems conditions may include abnormal quantities, high frequency of interrupts, etc.

12.3 Testing Checklist

This checklist is provided to assist you in understanding the testing issues of your project. If you cannot answer a question affirmatively, you should carefully examine the situation and take appropriate action.

12.3.1 Before Starting

- ❑ 1. Is testing planned for and considered throughout the entire development life cycle?
- ❑ 2. Is the overall testing strategy defined and documented, and is it an integral part of and consistent with the development program? [4]
- ❑ 3. Is the testing process well defined, documented, understood, and supported by the development team and management?
- ❑ 4. Are test requirements clearly defined? [4]
- ❑ 5. Are test methods, techniques, controls, and standards clearly defined and consistent with the testing strategy? [4]
- ❑ 6. Is each test activity traceable to specific requirements? [4]
- ❑ 7. Are configuration management and quality assurance in place and are they adequate to support the testing strategy? [4]
- ❑ 8. Are testers trained, skilled, and motivated people?
- ❑ 9. Have adequate time and resources been reserved for testing?

- ❑ 10. Are time and resources allocated for test preparation early in the project life cycle?

12.3.2 During Execution

- ❑ 13. Is testing used as a primary tool to ensure good project health?
- ❑ 14. Is testing implemented as a tool for improving product quality and the development process as a whole?
- ❑ 15. Is early life cycle testing used to prevent propagation of defects to later stages of development?
- ❑ 16. Is a tracking system being used to record what has been tested and what has not?
- ❑ 17. Is a database of test results being maintained for current and future reference?
- ❑ 18. Are tests used as milestone and progress indicators?
- ❑ 19. Is the right amount of testing being done to balance risk with available time and resources?
- ❑ 20. Are you using inspections and other evaluation methods (see Chapter 11) to reduce the errors found through testing?
- ❑ 21. Do you know when your testing is complete?

12.4 References

[1] Software Program Managers Network, *Little Book of Testing, Vol. 1*, 1998: www.spmn.com/products_guidebooks.html

[2] Jorgensen, Paul C., *Software Testing A Craftsman's Approach*, CRC Press, 1995, p.3.

[3] Kit, Ed, *Software Testing in the Real World*, Addison-Wesley, 1995, p.3.

[4] Software Program Managers Network, *Little Book of Testing, Vol. II*, 1998: www.spmn.com/products_guidebooks.html

[5] *Program Manager's Guide for Managing Software*, 0.6, 29 June 2001, Chapter 11: www.geia.org/sstc/G47/SWMgmtGuide%20Rev%200.4.doc

[6] University of South Australia, Software Testing notes: http://louisa.levels.unisa.edu.au/se1/testing-notes/testing.htm

12.5 Resources

Crosstalk Magazine: www.stsc.hill.af.mil/crosstalk/

- "The Problem with Testing": www.stsc.hill.af.mil/crosstalk/2001/07/index.html
- "Maintaining the Quality of Black-Box Testing": www.stsc.hill.af.mil/crosstalk/2001/05/korel.html
- "Proven Techniques for Efficiently Generating and Testing Software": www.stsc.hill.af.mil/crosstalk/2000/06/wegner.html
- " Model to Assess Testing Process Maturity": www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1998/11/burnstein.asp
- Planning Efficient Software Tests": www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1997/10/planning.asp
- "Using Statistical Process Control with Automatic Test Programs": www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1998/08/statistical.asp
- "Engineering Practices for Statistical Testing": www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1998/04/statistical.asp
- "Using Inspection Data to Forecast Test Defects": www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1998/05/inspection.asp

Department of Energy (DOE) *Software Engineering Methodology*, Chapters 7-9: http://cio.doe.gov/sqse/sem_toc.htm

NASA, Recommended Approach to Software Development, Sections 7-9: http://sel.gsfc.nasa.gov/website/documents/online-doc.htm

Program Manager's Guide for Managing Software, 0.6, 29 June 2001, Chapters 10-11: www.geia.org/sstc/G47/SWMgmtGuide%20Rev%200.4.doc

Software Testing Stuff: www.testingstuff.com/

Software Testing Institute: www.softwaretestinginstitute.com/

SPMN Testing Guidebooks available for download at: www.spmn.com/products_guidebooks.html

SPMN 16 Critical Software Practices: www.spmn.com/16CSP.html

University of South Australia, Software Testing notes: http://louisa.levels.unisa.edu.au/se1/testing-notes/testing.htm

This page intentionally left blank.