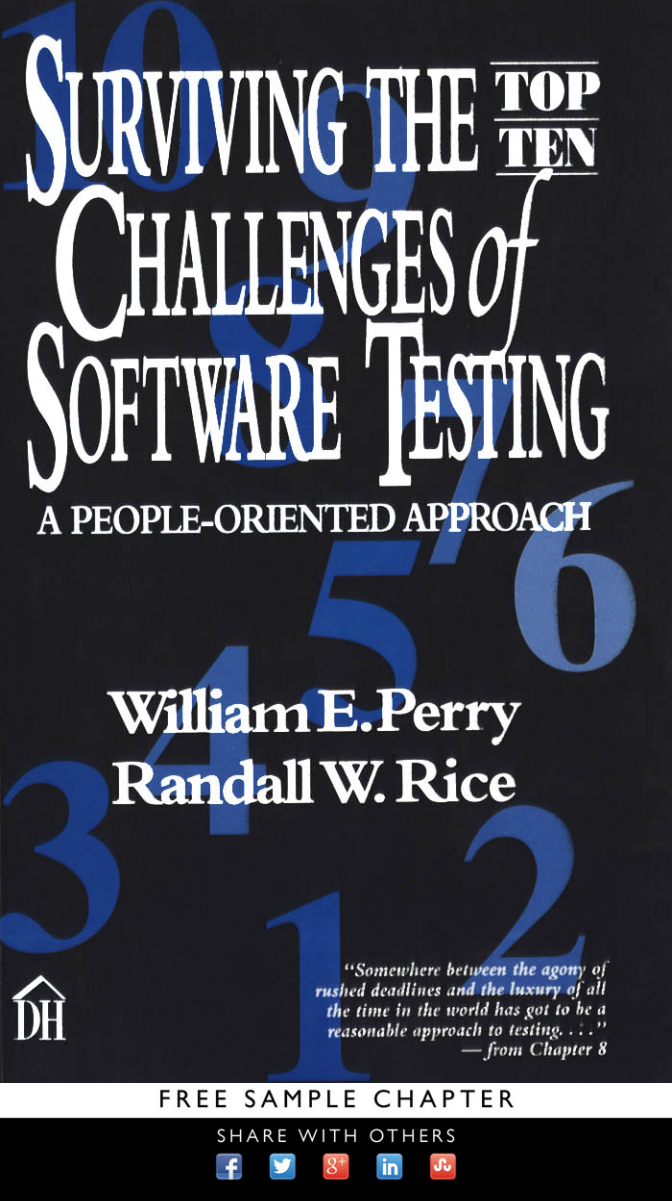
SURVIVING THE TOP TEN CHALLENGES of SOFTWARE TESTING

A PEOPLE-ORIENTED APPROACH

William E. Perry
Randall W. Rice



"Somewhere between the agony of rushed deadlines and the luxury of all the time in the world has got to be a reasonable approach to testing. . . ."
— from Chapter 8

SURVIVING THE TOP TEN CHALLENGES of SOFTWARE TESTING

A PEOPLE-ORIENTED APPROACH

Also Available from Dorset House Publishing

The Deadline: A Novel About Project Management
by Tom DeMarco
ISBN: 0-932633-39-0 Copyright ©1997 320 pages, softcover

Exploring Requirements: Quality Before Design
by Donald C. Gause and Gerald M. Weinberg
ISBN: 0-932633-13-7 Copyright ©1989 320 pages, hardcover

Handbook of Walkthroughs, Inspections, and Technical Reviews
by Daniel P. Freedman and Gerald M. Weinberg
ISBN: 0-932633-19-6 Copyright ©1990 464 pages, hardcover

Managing Expectations: Working with People Who Want More, Better, Faster, Sooner, NOW!
by Naomi Karten foreword by Gerald M. Weinberg
ISBN: 0-932633-27-7 Copyright ©1994 240 pages, softcover

Measuring and Managing Performance in Organizations
by Robert D. Austin foreword by Tom DeMarco and Timothy Lister
ISBN: 0-932633-36-6 Copyright ©1996 240 pages, softcover

Peopleware: Productive Projects and Teams
by Tom DeMarco and Timothy Lister
ISBN: 0-932633-05-6 Copyright ©1987 200 pages, softcover

Quality Software Management, Vol. 4: Anticipating Change
by Gerald M. Weinberg
ISBN: 0-932633-32-3 Copyright ©1997 504 pages, hardcover

Find Out More About These and Other DH Books

Contact us to request a Book & Video Catalog and a free issue of *The Dorset House Quarterly*, or to confirm price and shipping information.

DORSET HOUSE PUBLISHING
353 West 12th Street New York, NY 10014 USA
1-800-342-6657 212-620-4053 fax: 212-727-1044
dhpubco@aol.com www.dorsethouse.com

SURVIVING THE TOP TEN CHALLENGES of SOFTWARE TESTING

A PEOPLE-ORIENTED APPROACH

William E. Perry
Randall W. Rice

DORSET HOUSE PUBLISHING
353 WEST 12TH STREET
NEW YORK, NEW YORK 10014

Library of Congress Cataloging-in-Publication Data

Perry, William E.
Surviving the top ten challenges of software testing : a people-oriented approach / William E. Perry, Randall W. Rice.
p. cm.
ISBN 0-932633-38-2 (pbk.)
1. Computer software--Testing. I. Rice, Randall W. II. Title.
QA76.76.T48P493 1997
005.1'4--DC21 97-44565
CIP

Cover Design: David McClintock

Distributed in the English language in Singapore, the Philippines, and Southeast Asia by Toppan Co., Ltd., Singapore; in the English language in India, Bangladesh, Sri Lanka, Nepal, and Mauritius by Prism Books Pvt., Ltd., Bangalore, India; and in the English language in Japan by Toppan Co., Ltd., Tokyo, Japan.

Printed in the United States of America

Library of Congress Catalog Number: 97-44565

ISBN-10: 0-932633-38-2
ISBN-13: 978-0-932633-38-5

20 19 18 17 16 15 14 13 12 11 10

Digital release by Pearson Education, Inc., June, 2013

ACKNOWLEDGMENTS

We would like to thank Dorset House Publishing Company for recognizing testing as more than a technical problem. That made this book a reality. A special thanks goes to David McClintock, who did a superb job in editing our text and keeping the project on track, and to Wendy Eakin for her creative input and encouragement.

We would also like to thank the many professional testers and information technology professionals who have contributed to this book by participating in surveys, providing input during our seminars, and encouraging us to address the people-oriented issues in testing. Without this input, there would have been no basis for this book.

Finally, and most importantly, we would like to thank God for allowing us to be in a unique position at a unique time to help others with the message of this book. As the authors, we believe that God is the Master Tester. In our lives, God has tested us very extensively.

WILLIAM E. PERRY, CQA, CQE, CSTE
RANDALL W. RICE, CQA, CSTE

Dedication

To our wives and families,
whom we have tested on many occasions.

CONTENTS

SURVIVING THE TOP TEN CHALLENGES of SOFTWARE TESTING

A PEOPLE-ORIENTED APPROACH

This page intentionally left blank

CHAPTER THREE

CHALLENGE #10: GETTING TRAINED IN TESTING

OVERVIEW

A commonly held belief about testing is that anyone who can operate a system can perform testing. The truth is that testing is a professional discipline, requiring unique skills that are anything but intuitive. Without training, testers are ill-equipped to meet the rigors of testing, especially in technically difficult situations. The people-related challenge of this chapter is to secure adequate support for training.

Although testing training is available through a variety of sources, a commitment and investment to obtain training are always required of testers and management. In some organizations, this need is recognized and met. In other organizations, self-motivation is the only way to get the training required to build testing skills.

In this chapter, we look at ways to raise management's awareness of the training needs of testers. We also explain how to map out your personal skill-building goals and objectives, which may include certification as a software test engineer.

STATE OF THE PRACTICE

The project is two months away from completion and management realizes that testing should be performed. The only question is, who should perform testing? With everyone on the project already hopelessly involved with the daily battles at hand in building the system, who has time to perform testing?

The idea occurs to management that fresh, independent people would be a good way to test the project. After all, there are several people in the end-user area that could be pulled together for about four weeks. If the end-users need help, additional contract resources can always be secured at a fairly low hourly rate to pound the keyboards.

In fairly short order, the ad hoc test team is assembled and testing commences. As the deadline approaches, defects have been found and corrected. Everything looks pretty good to the test team. Then, the moment of truth arrives. The project that has cost the organization nearly one million dollars to complete is ready to be placed into production.

Day one of the new system arrives, complete with T-shirts and balloons. There is only one problem: After two hours, calls to the help desk start pouring in, reporting strange results. Before long, the system response time starts to slow down severely. Customers are irate because they can't get quick answers to their telephone inquiries. Finally, the inevitable happens—the proverbial plug is pulled. "Oops, let's hope we can restore the old system. . . ."

What Went Wrong in This Scenario?

This story illustrates a classic misunderstanding of the skills required in testing. In the project described, the testers had never tested software before and, furthermore, had never been trained to test. Although many mistakes were made in the organization of the testing effort, a major error was the assumption that anyone could perform testing and find critical defects.

Testers face a challenge in defining and executing test cases that cover even a fraction of the possible system functions. To understand the universe of tests that can be performed, a tester must be aware of the possible sources of defects. These sources of defects are not a deeply shrouded mystery, but they are part of a growing body of knowledge that trained testers possess.

Without adequate training in testing techniques, testers are left to their own devices. Some of their tests will find obvious defects but will miss the subtle defects that can cause system failure. Untrained testers often do not understand the features and functions of the software they are assigned to test.

Some companies start entry-level people in the testing area as a form of training and move them to the "more meaningful" jobs in software development after they learn how to use the system. In such situations, as with the scenario described above, support for training is not received by the testers because

- management is not aware of the value of testing
- organizations are not aware of training resources
- organizations see training as an extra expense that can be cut
- testers feel they are too busy to attend training sessions
- in-house trainers often lack testing expertise
- testing is perceived as something anyone should be able to perform

IMPACT ON TESTING

The testing effort can consume fifty percent or more of a project's total cost. At the same time, less than one percent of software professionals have been formally trained in testing techniques. This is a huge imbalance.

Without training, important test cases are overlooked, planning is either not completed or not performed at all, defect information is not tracked, and tests are performed several

times needlessly. In effect, testing becomes a craft rather than a repeatable process.

Training can help a tester fulfill his project responsibility and understand the kinds of testing that developers, technical support, and end-users usually perform. See Table 3.1. The intersection points in the table list the skills required to support testing.

Skill Categories and Descriptions

Below, the testing skills listed in Table 3.1 are divided into two categories: essential and optional. The essential skills are critical for effective testing. The optional skills add efficiency and optimize the testing process.

Essential Testing Skills

Test Planning: Analyzing a project to determine the kinds of testing needed, the kinds of people needed, the scope of testing (including what should and should not be tested), the time available for testing activities, the initiation criteria for testing, the completion criteria, and the critical success factors of testing.

Test Tool Usage: Knowing which tools are most appropriate in a given testing situation, how to apply the tools to solve testing problems effectively, how to organize automated testing, and how to integrate test tools into an organization.

Test Execution: Performing various kinds of tests, such as unit testing, system testing, user acceptance testing, stress testing, and regression testing. This can also include how to determine which conditions to test and how to evaluate whether the system under test passes or fails. Test execution can often be dependent on your unique environment and project needs, although basic testing principles can be adapted to test most projects.

Defect Management: Understanding the nature of defects, how to report defects, how to track defects, and how to use the

information gained from defects to improve the development and testing processes.

Optional Skills

Risk Analysis: Understanding the nature of risk, how to assess project and software risks, how to use the results of a risk assessment to prioritize and plan testing, and how to use risk analysis to prevent defects and project failure.

Test Measurement: Knowing what to measure during a test, how to use the measurements to reach meaningful conclusions, and how to use measurements to improve the testing and development processes.

Test Case Design: Understanding the sources of test cases, test coverage, how to develop and document test cases, and how to build and maintain test data.

Building a Test Environment: Knowing how to create a test environment based on the technical requirements, how to implement configuration control for the test environment, and how to maintain the test environment.

Knowing What Kinds of Testing Should Be Performed

Table 3.1 shows three attributes of system development commonly subject to testing: function, structure, and quality. Functional testing tests the software from a cause/effect perspective. For example, if a certain function key is pressed, is the result correct? Other examples could include adding a new customer, opening a window on a GUI application, or updating data.

Structural testing requires a knowledge of the software code or system internals. This kind of testing seeks to test the logic and interfaces of a system. Examples of this kind of test would be testing each branch of an IF statement in a software module, or performing a stress test to determine how many concurrent users the system can handle.

Table 3.1.
Testing Skills Needed.

Who Tests? / What Do They Test?	System Functionality	System Structure	System Quality
Developers	Test Planning, Test Tool Usage, Test Execution, Risk Analysis, Test Case Design, Building a Test Environment	Test Planning, Test Tool Usage, Test Execution, Test Case Design, Building a Test Environment	Test Planning, Test Tool Usage, Test Execution, Risk Analysis, Test Case Design, Building a Test Environment
Independent Testers	Test Planning, Test Tool Usage, Test Execution, Defect Management, Risk Analysis, Test Measurement, Test Case Design, Building a Test Environment	Test Planning, Test Tool Usage, Test Execution, Test Case Design, Test Measurement	Test Planning, Test Tool Usage, Test Execution, Defect Management, Risk Analysis, Test Measurement, Test Case Design, Building a Test Environment
Technical Support	Does not usually perform	Test Planning, Test Tool Usage, Test Execution, Test Case Design, Building a Test Environment	Test Planning, Test Tool Usage, Test Execution, Test Case Design, Building a Test Environment
End-Users	Test Planning, Test Tool Usage, Test Execution, Defect Management, Risk Analysis, Test Case Design	Does not usually perform	Test Planning, Test Tool Usage, Test Execution, Defect Management, Risk Analysis, Test Case Design

System quality testing tests the things that a system should *be* as opposed to what the system should *do* (functional tests). Examples of this kind of testing would be usability testing for ease of use, performance testing for system performance, and reliability testing.

A tester needs to know the kinds of testing that should be performed on a project. With all the different kinds of projects under way in organizations, how is a tester supposed to know—without training—the most effective test methods? For example,

- client/server systems have a different set of testing concerns than traditional systems
- graphical user interfaces require a different approach than procedural on-line software
- user acceptance testing uses a different method of testing than system testing

Table 3.1 shows the skills needed for each kind of test, depending on the role of the tester: developer, independent tester, technical support, or end-user. After determining the testing skills needed, you can use Table 3.2 to find the best kinds of training. Here, each of the testing skills listed in Table 3.1 is related to training sources, such as seminars, conferences, books, and more, and is rated by degree of fitness based on the following scale:

Good Fit [1]: The training addresses specific issues that testers might face. Specific questions can be asked of and answered by the resource.
Acceptable [2]: The topic is discussed, but not to the degree that would meet everyone's needs. Questions may or may not be answered.
Marginal [3]: Some useful information will be learned, but application will likely be lacking. Questions are not answered.

Table 3.2. Types of Test Training Available.

Training Types / Skill Types	Test Planning	Test Tool Usage	Test Execution	Defect Management	Risk Analysis	Test Measurement	Test Case Design	Building a Test Environment
Seminars	1	3	1	1	1	1	1	1
Conferences	1	2	1	1	2	1	1	1
Vendor courses	2	1	1	1	3	3	2	1
Books	2	3	2	3	2	1	1	3
Videos	2	3	2	2	2	2	2	3
Self-study courses	2	3	3	3	3	2	2	3
In-house developed courses	1	1	1	1	2	1	1	3

1 = Good Fit, 2 = Acceptable, 3 = Marginal

Books contain details and can be referenced, but the author is not available to answer questions. Seminars contain information at a detailed level, but this must be applied to a variety of environments. Fortunately, if you have a question, the instructor should be able to answer it. Conferences are great places to learn what others are doing and what testing experts advise. A conference is not intended to provide a classroom experience, so there is usually a lot of interaction with peers and experts to learn—at least on a superficial level—new testing techniques.

The skill types introduced in Tables 3.1 and 3.2 can be used to answer the what, who, when, and how questions of testing.

What to Test
(Skills Required: Test Planning, Test Case Design, Risk Analysis)

A common question that testers pose is "What do we test?" That is an understandable question in light of the billions of possible test conditions.

The quality and extent of the testing effort depends on how well testing identifies and mitigates risk, and how much of the software function and structure are tested (coverage). Tests that meet these criteria are not created by spur-of-the-moment inspiration—they must be planned in advance. To understand how to write a test plan requires training in testing fundamentals, much like developing a data model requires training in the science of data modeling.

Without training, the tester is working at minimal effectiveness because he or she is not aware of the wide range of possible sources of test cases. Except in the simplest of cases, it is impossible to test all of the possible combinations of software functions. However, it is possible to identify critical software functions and develop a set of test cases that cover those functions, provided the tester knows how and where to find those cases.

Who Should Perform Testing (Skills Required: Test Planning)

In contrast to the story at the beginning of this chapter, everyone should have a part in testing a project: developers, independent testers, end-users, and management. The question is, in which ways do these people contribute value to the testing effort? Certain roles facilitate certain types of testing more effectively:

- Developers are best suited to perform code-based tests at the unit level.
- Developers and/or independent testers are best suited to perform unit testing.
- Combinations of people with a variety of talents are often needed to test at the system level.
- Users should own the user acceptance testing effort.

If people are not matched to the appropriate type of test, a testing effort will prove ineffective. When users are told to test software at a unit level, there is a waste of time and talent. Unit testing involves testing each and every edit and function. When users get bogged down in detail, they lose their most important contribution to the testing effort—the business perspective. Users should be most concerned with validating that the system will support the needs of the organization. Again, Table 3.1 shows which project roles are best suited for the different kinds of testing and which skills are needed for effective testing.

When Testing Should Be Performed (Skills Required: Test Execution, Test Planning)

Another major mistake made in the testing example was waiting until the end of the project to test the system. A trained tester would know that testing starts at the beginning of the project, with verification of system deliverables such as

requirements documents, and continues throughout the life span of the system. Training in test execution and planning provides knowledge about the proper time for each phase of testing during a project.

One of the best ways to visualize the phases of testing and when they should be performed is the "V" diagram (see Figure 3.1). The "V" diagram shows the order of development activities along with the corresponding order of testing activities. In the case of a major system development project, work starts at the upper left of the "V" with defining the business or organizational need. Requirements are developed and the system is designed and built. In rapid application development (RAD), these activities are usually performed in cycles. You will notice that the requirements step involves two major kinds of tests: verification (inspections, walkthroughs, and other reviews) and validation (system testing). This same idea is applied to each development step to find defects throughout the project.

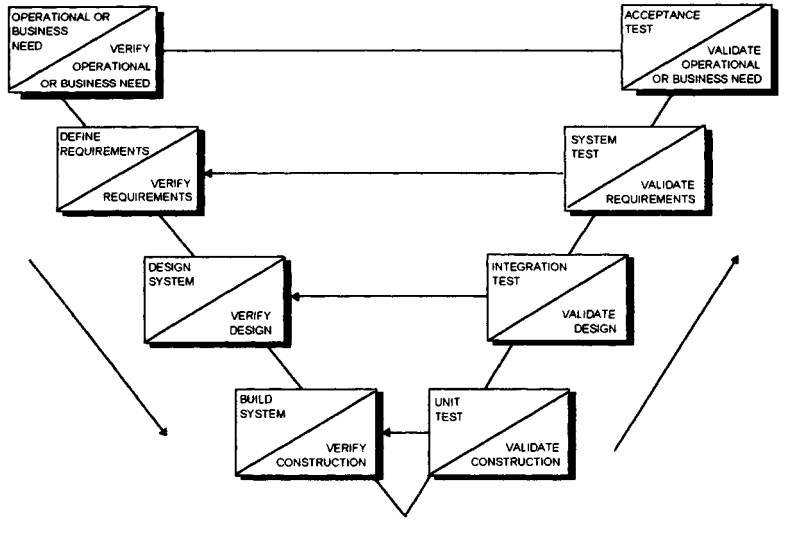


Figure 3.1: The "V" Diagram.

How to Test
(Skills Required: Test Execution, Defect Management, Test Measurement, Building a Test Environment)

A trained tester will know, among other things, how to write a test plan, how to create test cases, how to build an adequate test environment, how to perform a test, and how to evaluate the results of testing. These are the testing tasks that most people take for granted but that actually require a great deal of skill to perform. Without training, testers tend to reinvent the work of others through trial and error. The problem is that trial-and-error testing costs time and money that few projects can afford.

SOLUTIONS TO THE CHALLENGE

Of all the testing challenges, training is the easiest but not necessarily the least costly to address. As we discussed above, training can take the form of books, seminars, conferences, and even magazine articles and learn-at-lunch sessions (provided someone has the in-house testing expertise to teach them). The form of training you should obtain depends on

- the availability of the subject material in that format
- the degree of fitness for the skills needed (see Table 3.2)
- the amount of time budgeted for training
- the amount of money budgeted for training
- the level of in-house testing expertise

Raise Management Awareness of Testing

Without management support, the development of testing skills will be very slow, if they develop at all. The key is for management to understand how much time and money is wasted on trial-and-error testing. Management also needs to understand that expensive and high-risk systems are entrusted to testers to ensure that defects do not occur when used by end-users or customers. It's management's choice: Who

would you rather have in this critical role, a trained and skilled tester or someone who is doing the best he can with the best ideas he can develop on his own?

Here are some steps you can take to raise management's awareness of testing and the need for skill-building:

1. *Calculate how much testing costs your organization.* Don't forget to include the cost of planning and reporting.
2. *Educate management in the cost of testing.* Show ways that the cost can be reduced through the use of more effective techniques. Most organizations can quickly reduce the cost of development and testing by at least ten percent by eliminating the waste and redundancy found in ad hoc testing processes.
3. *Explore other motivations for the benefits of testing.* For example, effective testing will help streamline your testing process and shorten the time to delivery.
4. *Build testing skill development into your personal goals and objectives.* Demonstrate to your management that you are committed to learning more about testing to add value to the organization. This means you might have to buy books on your own, but a little initiative goes a long way.
5. *Keep your eye out for articles about testing.* Route these articles to your management as you find them, to highlight the contribution skilled testers make on critical projects.
6. *Discuss your training needs with your management.* Explore ways together that you can find answers to your questions. If there is a relevant course available, discuss the possibility of attending and reporting the information back to the rest of the team.
7. *Find out if there is a training budget for your organization.* If there are funds available, it might be feasible to schedule an in-house training course for twenty or more people at a reasonable price.
8. *Be creative.* Buy another copy of this book, place a paper clip at this chapter, and leave it in your manager's in-basket as an anonymous gift!

Make Time for Training

The paradox of training is that when you need it the most, there isn't enough time. You'll want to put training into practice as soon as possible, but how do you make the time for training in the first place, when there is hardly time for testing? Here are some steps for getting the time for training:

1. *Build time for training into the project plan.* This makes training a planned event, not something extra to be squeezed in as time allows. Figure 3.2 shows a sample test plan that addresses training. Notice that the sample test plan is performed at a checkpoint. In a typical project, you would have a document like this for each project checkpoint, such as requirements definition, design, system testing, and so forth.
2. *Budget for training activities.* This will allow you to plan in advance which training you can secure.
3. *Match the training to your need.* To learn about many aspects of testing, you may require a week-long course. For a specific aspect of testing, a one-day course may suffice.

Develop Your Own Skills

As a tester, you have the ultimate responsibility for developing your skills. If your management supports your efforts in the form of paying for your training, that's great. If not, then you will need to find ways to develop skills that can fit into your personal budget. There are many tangible ways to develop your own testing skills:

1. *Make a list of your personal development goals.* These goals should include completion dates and what it will take on your part to achieve them.
2. *Build your own testing library.* Start with the books that address basic testing topics. A list of these are found in the Related Reading section at the back of this book.

PROJECT: OMNI	**CHECKPOINT:** System Testing
SCHEDULE/DATES	
Plans:	System test plans to be completed by September 30.
Training:	Training in testing techniques to start on September 7 and finish on September 14.
Testing Materials:	1 work room with a whiteboard, 4-drawer filing cabinet, and 2 telephones. Executable versions of the OMNI software.
Start Test:	October 15
Conclude Test:	November 15
BUDGET	
	$20,000
RESOURCES	
Equipment:	15 IBM-compatible 486 PCs attached to the local area network and 2 laser printers.
Software/ Documents:	Executable versions of the OMNI software, requirements documents for the OMNI system, OMNI system documentation and user guides.
Personnel:	3 developers from the OMNI project team, 1 quality assurance analyst, 3 representatives from each of the affected end-user departments, 1 internal auditor for testing the payroll subsystem.
TESTING MATERIALS	
System Documentation:	OMNI system requirements, data flow diagrams, entity-relationship diagrams, business process workflow diagrams, screen layouts, listing of all modules in the OMNI system.
Software to Be Tested:	All modules in the OMNI system, plus interfaces to the payroll and accounting systems.
Test Inputs:	Test cases as designed and documented by the test team.
Test Documentation:	Test scenarios, test scripts, and the system test plan. Defect reports from the system test.
Test Tools:	Capture/playback, defect tracking tool, test case generator.

Figure 3.2: System Checkpoint Administrative Worksheet (Part 1).

TEST TRAINING	
Type of Training:	Training in software and system testing techniques.
Personnel to Be Trained:	Project manager, QA analysts, end-users, internal auditor.
When:	September 7 to 14
Where:	Corporate training facility.
Training Staff/Vendor:	In-house testing course conducted by in-house training staff.
Training Materials:	Notebooks, overhead projector, VCR, and monitor.
TESTS TO BE CONDUCTED	
See attached list of test descriptions.	

Figure 3.2: System Checkpoint Administrative Worksheet (Part 2).

3. *Keep a clipping file.* This should include testing articles and other articles of interest that appear in computer trade magazines and other publications.
4. *Participate in local software quality organizations.* A listing of these groups appears in the back of this book.
5. *Continue the learning process.* As technologies change, so must your tool kit of techniques.

Certify Your Testing Skills

As you gain experience and training, you should apply to become a Certified Software Test Engineer (CSTE). This will add to your credibility and enhance your testing career. It tells your employer and others that you have developed the skills and obtained the experience to test software effectively. In addition, certification is something that you can carry to your next job to show evidence of demonstrated testing skills.

Your author Bill Perry's organization, the Quality Assurance Institute (QAI) of Orlando, Florida, has a certification program for people with two or more years of professional testing experience. Details on how to contact QAI are provided in the Resources section of this book.

SOLUTION IMPEDIMENTS

If you really want to build your skills in testing, no one can keep you from learning and refining them. Along the road of learning, you will likely face impediments that will require creative solutions. Many of these present themselves as "what if" or "how do I" questions, as we'll see throughout this book.

What if my management will not support my training needs?

Even if your current management does not support your training needs, there is hope. Things change and your situation a year from now can be greatly different from what it is today—that is why it is so important not to wait for management to

start your skill-building process. Buy or check out the books you need to read, network with testers in other companies, and generally do what it takes to learn on your own. If your current situation does not change, you might choose to take your skills to an employer who will recognize and reward your initiative.

What if I do not have enough money to attend a testing seminar or conference?

If you're on your own to get training, courses can get expensive. Take advantage of every opportunity to network with other testers, attend local presentations on testing by test tool vendors, and check out testing sources on the Internet. In addition, many local libraries have books on testing that can start you on the road to gaining testing knowledge.

What if I do not have enough time for training?

The urgent often takes the place of the important. Ask yourself this question: "If I really knew what I was doing, would it take as long?" Probably not. Sometimes you have to make time to do the truly important things. Training is an investment that pays big dividends in both efficiency and effectiveness.

GUIDELINES FOR SUCCESS

- **Set personal goals.**

  If you set personal goals, you will be in a three percent minority of successful people. Success seldom happens without planning.

- **Seek management's assistance in building your testing skills, but don't wait for it.**

  Enlightened managers invest in people and reap dividends in prevented problems. Unenlightened managers focus on

cutting costs, do not provide the tools to do the job right, and will instead spend much time and money on fixing problems.

- **Invest in yourself.**

  "What you become directly influences what you get"
  —Jim Rohn

- **Strive to add value to the testing effort.**

  Job security in America is a thing of the past. Companies are looking for the people who add value. This is the case whether you work for yourself or for someone else. A trained tester adds value to a project by knowing and applying sound testing methods. Effective testing methods can help eliminate waste and redundancy while increasing the number of defects found.

PLAN OF ACTION

While we have addressed solutions and impediments, let's look at an overall plan of action that ties everything together into your master approach for building testing skills.

1. Get management on your side by showing them

   - how much money is being spent on testing
   - the critical nature of software quality to your organization—that is, what could happen if a defect is found by users or customers
   - how much time and money could be saved by applying effective testing methods
   - the value that you personally could bring to the organization as a result of increasing your skill
   - initiative by starting to build your own skills

2. Develop your own skill-building goals and objectives by

    - making a list of what you need to learn to be effective
    - identifying resources for training and skill-building in testing
    - planning to spend the time and money it takes to meet your skill-development goals
    - staying on the lookout for training opportunities that you can easily participate in, such as local special interest groups, vendor test tool presentations, and the like
    - getting certified as a Certified Software Test Engineer

3. Never stop learning!

INDEX