Volume 5, Issue 8
August 2002—The Software Quality Advisor

The Software Quality Advisor Online

Are Use Cases a Substitute for Requirements?

Randall W. Rice, CSQA, CSTE

Are Use Cases a Substitute for Requirements?

This article explores use cases and their relationship to requirements.

- What are use cases and what is their best application on a project?
- What is a requirement?
- Why do people want to substitute use cases for requirements?
- Can use cases be effectively substituted for requirements?

Use cases in software development are becoming very popular. One question I get repeatedly as I teach our seminars on Gathering, Documenting and Testing User Requirements is "Can use cases be substituted for user requirements?"

A Little Background

First, let's look at what use cases are and examine a little about their origin. It's important to get a good definition of terms, so for the term "use case," I went to one of the best sources I could think of, a paper written by Dean Leffingwell of Rational Software. Leffingwell defines a use case in the following terms:

"Technically, a use case describes a sequence of actions, performed by a system, that yields a result of value to the user."[1]

In the Unified Modeling Language, one of the models is a use case model, which shows various actors (users) and their relationship with processes described by use cases.

In most books and articles on use cases, the use case focuses on a sequence of actions from a particular actor's perspective. An expanded view of use cases may also include pre-requisites, expected results, exceptional conditions and alternate courses of action.

The great application of use cases is that they provide an effective format to communicate to developers how a user will perform a particular function in the context of the application.

A sample use case is included at the end of this article.

Contrast the idea of a use case with that of a Software Requirements Specification (SRS). A requirement can be defined as:

- *a software capability needed by the user to solve a problem that will achieve an objective, or*
- *a software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documentation.*[2]

(Continued on page 2)

Inside this issue:

Book Review—*Introducing Software Testing* by Louise Tamres

Overview

Introducing Software Testing is a good treatment of many techniques that have been successfully used by testers over the years. This book has a good philosophy behind it that says requirements and processes are important in testing. However, the book presents the information in a way that people in organizations that may not have firmly defined requirements or processes can still easily apply the techniques described in the book.

What I Liked About the Book

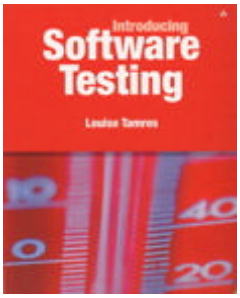
There was a lot for me to like about this book.

- ***A wide coverage of test and test-related topics***

There are many topics in software testing – just go to a testing conference to see some of the possibilities. Tamres does a good job in covering the major topics in a way that leaves the reader with an understanding of what's needed for software testing.

- ***Appropriate for testers at all levels***

Although the title is "Introducing Software Testing," I would not dismiss it too quickly as being a book just for beginners. I have been a full-time tester and trainer in testing for 14 years and still learned valuable things from this book.



★★★★★

buy it now

(Continued on page 2)

Book Review—*Introducing Software Testing* by Lousie Tamres

(Continued from page 1)

- ***Adequate detail***

I never left any topic asking "why?" or "how?". The level of detail is a good balance between readability and having enough detail to explain the topics.

- ***Lots of examples***

For every topic, there are multiple examples of how to apply the techniques described in the book. These examples show realistic ways the techniques would be applied in an actual project.

- ***Proven and practical techniques***

The techniques shown in the book are the same ones that testers have been using for many years, but having them in one book is a great thing. The techniques are easy to understand and apply. I'm all for creative and new techniques for testing, but for foundational approaches that have been proven to work and to scale for large projects, I like to keep firmly grounded in good practices.

Audience

- Beginning Testers
- Experienced Testers
- Trainers
- Technical Managers

Outtakes

"A project is in panic mode and the deadline is rapidly approaching. Management starts to think about the need to test this product, having already missed some prime opportunities for improving software quality. One unfortunate programmer is assigned the task of software testing, which is often viewed as being transferred to purgatory. Needless to say, this poor hapless soul is given no guidance, and nobody in the organization is capable of providing any help. Despite the poor condition of requirements and other product documentation, the product is being built and it will be shipped. The task given to the tester is to minimize the surprises that could manifest themselves after the product is installed at customer sites. Under extreme pressure, this untrained tester is very inefficient and is at a loss how to begin. A clueless manager may even purchase testing tools, despite there being no useful tests to automate. This is the scenario that gives software testing a bad name.

Software testing is a specialized discipline requiring unique skills. Software testing is not intuitive; one must learn how to do it. Naïve

(Continued on page 5)

Use Cases

(Continued from page 1)

The IEEE standard 830 is used to provide a full-featured standard for documenting requirements. The IEEE standard is very complete and has been used in a wide variety of applications. In addition to functionality, a standard may also address interfaces, performance, usability and testability. Use cases do not address these aspects of an application.

Where's the Beef?

My observation is that software developers, and to some degree users as well, have historically tried to avoid documenting requirements. After all, they are difficult to gather and document, plus they tend to change throughout a project. However, no matter which technology or methodology is in vogue, we always get back to the issue of dealing with the description of the solution so we will know what to build, when the solution has been successfully delivered, and how to test the solution.

Use cases have given hope to people seeking to describe what is to be built without writing requirements. In some ways, people have tried to employ use cases as a middle ground approach that is not as detailed and structured as a requirement, yet more than no documentation at all. However, the traditional use of use cases is to describe a user's process, not the product. This leads me to ask how far can use cases be extended before they are no longer use cases, but a hybrid requirement? In addition, is a hybrid document necessary? Why not just write a requirement document to describe features and use cases to describe processes?

This issue goes beyond use cases and requirements. We need to examine what really drives software development. I like the way that Dr. Timothy Korson states it,

"Good software engineering is NOT use case driven! Requirements are important. Use cases are a good way to structure requirements, but if you care at all about component based development, reuse, robust distributed architectures, cost, schedule, etc., than you cannot afford to let any single viewpoint drive your project." He goes on to say,

"Good software engineering is driven by a number of concerns that are weighted differently by different organizations and different projects within an organization. These concerns include: technical design considerations, user requirements, reuse, modifiability, performance,

(Continued on page 3)

"This leads me to ask how far can use cases be extended before they are no longer use cases, but a hybrid requirement?"

Use Case Model

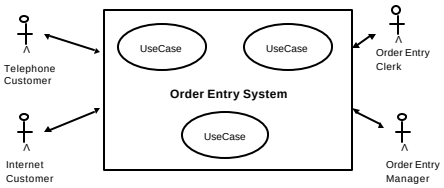


Figure 1—Sample Use Case Model Structure for an Order Entry Application

Use Cases

(Continued from page 2)

standardization concerns, schedule pragmatics, and other business drivers."[3]

Creative Uses for Use Cases

People have been able to creatively apply use cases for some very natural extended purposes, such as test scripts and test cases, as well as user documentation. Anything that requires the documentation of a process can benefit from use cases as input.

Summary

I don't believe that use cases are effective substitutes for requirements, or are the drivers of software development. Use cases are very helpful for conveying how someone will use an application, and this documentation can extend to helping testers design tests, as well as other purposes beyond building the application.

However, use cases fall short of being an effective vehicle for describing complex business rules, performance requirements and other non-process aspects of an application. As much as people may want to find a way to develop software without documented requirements, I just don't see it happening soon.

There should be a balance of well-defined requirements and use cases that provide the usage context for an application. These two effective documents, plus other models and techniques, such as prototyping, can provide a clear picture of what is to be built and how to test it.

1. Leffingwell, Dean. Features, Use Cases, Requirements, Oh my! www.rational.com
2. Dorfmann, Merlin, and Richard H. Thayer. *Standards, Guidelines, and Examples of System and Software Requirements Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1990.
3. Korson, Timothy. Constructing Useful Use Cases - http://www.software-architects.com/publications/index.html

Sample Use Case

Name: Web-based Order Entry
Actors: Internet Customer
Objective: To place orders via the internet from the company's web site.
Preconditions: The following data must be available to place and complete the order:

Customer data
- name
- address
- phone
- physical address
- e-mail address

Product data
- description
- price
- weight
- discounts
- restrictions (customs, age, state, tax, shipping)

Results (Post-conditions)

Completed order transaction, financial transaction and fulfillment transaction
Updated customer data

Detailed Description

Seq.	Action	Expected Result
1.	Customer accesses web site for the first time	Company home page appears correctly
2.	Customer searches for product	Products are easy to find. Product information is helpful and complete
3.	Customer selects product by adding to shopping cart	Item added to shopping cart correctly. Shopping cart total is maintained correctly.
4.	Customer proceeds to checkout	Checkout page displayed correctly with correct total.
5.	Customer enters name, address, phone number, e-mail address, and credit card information.	Customer information accepted correctly. Edits screen out invalid information.
6.	Customer selects desired shipping method	All available shipping methods are displayed correctly. Shipping totals are correctly computed, based on total weight, customer's address, and business rules for discounts.
7.	Customer finalizes order and confirms total	Credit card is authorized for payment. Confirmation page displayed to customer. Order transaction sent to order entry system. Fulfillment transaction sent to warehouse. Confirmation e-mail is sent to customer.

Exceptions
- Incorrect zip code
- Incorrect credit card
- Product ordered is not in stock
- Foreign customer
- Preferred customer
- Government customer

Alternate Courses
Customer desires to send check instead of credit card information.
Customer cancels order while in checkout.
Customer wants to cancel order after checkout confirmation.

Links

The Computer Security Resource Center (CSRC) at the National Institute of Standards and Technology has a great list of information, standards and checklists for all types of computer security at: http://csrc.nist.gov/publications/nistpubs/index.html

Need templates for security planning? Here's the place to get them! Infosyssec – Security Portal at: http://www.infosyssec.org/infosyssec/secpol1.htm

Seven Deadly Security Sins
At: http://www.newsfactor.com/perl/story/19116.html

The following links are helpful if you are trying to find information to add to your knowledgebase on software testing:

Tutorial on Software Testing

http://www.elet.polimi.it/Users/DEI/Sections/CompEng/Mauro.Pezze/Tutorials/ICSE98/icse98/

Case Studies of Software Failures

http://www.internetwk.com/story/INW20020821S0005

Software Testing Techniques – Test Case Design

http://louisa.levels.unisa.edu.au/se1/testing-notes/testing.htm

A Tutorial of Test Design

http://www.stagsoftware.com/images/TestingTutorial.pdf

White Paper on Unit Testing

http://www.mobilein.com/WhitePaperonUnitTesting.pdf

A Review of Testing

http://ls.afit.edu/spdp/CSE495/Slides/Final%20Review%20v1.ppt

Software Quality Tutorial

http://www.eelab.usyd.edu.au/takeme/secourse/m6.ppt

Software Testing Techniques

http://snowhite.cis.uoguelph.ca/~nonnecke/courses/cis343w02/lecturenotes/Testing.pdf

Text Summary of The Art of Software Testing

http://www.ncf.carleton.ca/~dm720/softest.html

Quotes

"Be at war with your vices; at peace with your neighbors, and let every new year find you a better man."
- Benjamin Franklin (1706 - 1790)

"If in the last few years you have n't discarded a major opinion or acquired a new one, check your pulse. You may be dead."
- Gelett Burgess (1866 - 1951)

"Good resolutions are like babies crying in church. They should be carried out immediately."
- Charles M. Sheldon (1812 - 1904)

"No one has more time than you have. It is the discipline and stewardship of your time that is important. The management of time is the management of self; therefore if you manage time with God, he will begin to manage you."
- Jill Briscoe

"Next to the dog, the wastebasket is your best friend."
- B.C. Forbes

"Those who know how to win are much more numerous than those who know how to make proper use of their victories."
- Polybius

"I do not believe in a fate that falls on men however they act; but I do believe in a fate that falls in them unless they act."
- G.K. Chesterton

"Wealth, like happiness, is never attained when sought after directly. It always comes as a by-product of providing a useful service."
- Henry Ford

He who seeks good finds goodwill, but evil comes to him who searches for it.
- The Bible, Proverbs 11:27

Questions From the e-Mail Bag

Q: The Quality Managers at my company are struggling to move Quality forward in the SDLC, or, to be honest, to just get it moved into place anywhere! Each time we bring up quality principles and practices, the same questions arise. While I've searched your web site, along with several others, I'm unable to find an answer: any help you can offer would be appreciated.

There are two primary questions:

1 - On an average project in a CMM level one company, what percentage of testing is caused by rework?
2 - Again, on an average project in a level one company, what percentage of defects is caused by failure in requirements (requirements being unclear, inconsistent, incomplete, incorrect, inconcise, unambiguous, or untestable.)?

Any help you could offer would certainly help us move our processes forward.

A: You raise a couple of interesting questions. I can answer the second one better than the first.

On the testing driven by rework, I don't have any research or metrics on that. There are a lot of factors that impact post-implementation rework and testing, such as poor configuration management and version control, lack of adequate testing, incom-

(Continued on page 5)

Links

The Computer Security Resource Center (CSRC) at the National Institute of Standards and Technology has a great list of information, standards and checklists for all types of computer security at: http://csrc.nist.gov/publications/nistpubs/index.html

Need templates for security planning? Here's the place to get them! Infosyssec – Security Portal at: http://www.infosyssec.org/infosyssec/secpol1.htm

Seven Deadly Security Sins
At: http://www.newsfactor.com/perl/story/19116.html

The following links are helpful if you are trying to find information to add to your knowledgebase on software testing:

Tutorial on Software Testing

http://www.elet.polimi.it/Users/DEI/Sections/CompEng/Mauro.Pezze/Tutorials/ICSE98/icse98/

Case Studies of Software Failures

http://www.internetwk.com/story/INW20020821S0005

Software Testing Techniques – Test Case Design

http://louisa.levels.unisa.edu.au/se1/testing-notes/testing.htm

A Tutorial of Test Design

http://www.stagsoftware.com/images/TestingTutorial.pdf

White Paper on Unit Testing

http://www.mobilein.com/WhitePaperonUnitTesting.pdf

A Review of Testing

http://ls.afit.edu/spdp/CSE495/Slides/Final%20Review%20v1.ppt

Software Quality Tutorial

http://www.eelab.usyd.edu.au/takeme/secourse/m6.ppt

Software Testing Techniques

http://snowhite.cis.uoguelph.ca/~nonnecke/courses/cis343w02/lecturenotes/Testing.pdf

Text Summary of The Art of Software Testing

http://www.ncf.carleton.ca/~dm720/softest.html

Quotes

"Be at war with your vices; at peace with your neighbors, and let every new year find you a better man."
- Benjamin Franklin (1706 - 1790)

"If in the last few years you have n't discarded a major opinion or acquired a new one, check your pulse. You may be dead."
- Gelett Burgess (1866 - 1951)

"Good resolutions are like babies crying in church. They should be carried out immediately."
- Charles M. Sheldon (1812 - 1904)

"No one has more time than you have. It is the discipline and stewardship of your time that is important. The management of time is the management of self; therefore if you manage time with God, he will begin to manage you."
- Jill Briscoe

"Next to the dog, the wastebasket is your best friend."
- B.C. Forbes

"Those who know how to win are much more numerous than those who know how to make proper use of their victories."
- Polybius

"I do not believe in a fate that falls on men however they act; but I do believe in a fate that falls in them unless they act."
- G.K. Chesterton

"Wealth, like happiness, is never attained when sought after directly. It always comes as a by-product of providing a useful service."
- Henry Ford

He who seeks good finds goodwill, but evil comes to him who searches for it.
- The Bible, Proverbs 11:27

Questions From the e-Mail Bag

Q: The Quality Managers at my company are struggling to move Quality forward in the SDLC, or, to be honest, to just get it moved into place anywhere! Each time we bring up quality principles and practices, the same questions arise. While I've searched your web site, along with several others, I'm unable to find an answer: any help you can offer would be appreciated.

There are two primary questions:

1 - On an average project in a CMM level one company, what percentage of testing is caused by rework?
2 - Again, on an average project in a level one company, what percentage of defects is caused by failure in requirements (requirements being unclear, inconsistent, incomplete, incorrect, inconcise, unambiguous, or untestable.)?

Any help you could offer would certainly help us move our processes forward.

A: You raise a couple of interesting questions. I can answer the second one better than the first.

On the testing driven by rework, I don't have any research or metrics on that. There are a lot of factors that impact post-implementation rework and testing, such as poor configuration management and version control, lack of adequate testing, incom-

(Continued on page 5)

Questions from the Mail Bag (Continued from Page 4)

(Continued from page 4)

plete or missing user requirements, etc. I often go back to the overall cost of quality: prevention, detection and correction of defects. Also, I recognize that some companies do well even at CMM 1 - the issue is that level is how good are the people and are things under control, even without processes.

For years, I have heard that the "average" cost of quality for software is about 40% of the development costs. So, unless an organization can radically improve processes, quality improvement just shifts the percentages of prevention, detection and correction. My gut estimation is that in level 1 companies about 30 - 40% of ongoing maintenance changes have to have some level of post-implementation rework. I have seen it go as high as 80%. Using that estimate, I would guess about one-third of testing is caused by rework - IF you are in a maintenance environment. You can probably use that percentage for development projects as well, especially if there are bad requirements.

For the second question about the percentage of defects caused by defects in requirements, this has been measured for over 30 years and the numbers still track about the same. Approximately 50% of the defects on a project can be traced back to defects in requirements. Of course, if there are no written requirements, all defects could theoretically be traced back to non-documented requirements. BTW - in those same numbers about 25% of defects are traced back to design defects and only about 7% to coding errors.

Q: I am working on an ERP software development project at the customer end and we are about to start system acceptance test.

You suggested that a good approach to user acceptance testing is to evaluate the software package against how it is going to be used on a daily basis, I believe it quite normal to map these business processes and document them as "Business Scenarios" to reflect actual business usage. Would you agree that running acceptance tests against these "Business Scenarios" is a healthy test?

Of course these business scenarios should reflect the businesses SOP's and the acceptance test should also include:

. Help text

(Continued on page 6)

"True validation is based on real-world conditions, not paper-based specifications."

Book Review—Introducing Software Testing

(Continued from page 2)

managers erroneously think that any programmer can test software — if you can program, then you can test. This is the motivation behind this book: to provide a step-by-step approach for getting started with the testing effort."

—

"In the chaotic software development environment, adequate requirements are rarely provided, and if they are, their completeness and correctness are questionable. In a situation when no one has analyzed the requirements adequately, the burden falls on the tester to pursue requirements issues prior to defining any tests. It is often impossible to perform thorough testing, given the tight schedules and limited resources. It is possible, however, to make intelligent choices and maximize the effectiveness of the testing effort.

The goal is to learn how best to approach the testing tasks and eventually produce a workable test process for future projects."

—

"I do not advocate working from poor requirements. Properly analyzing requirements corrects many deficiencies. Reviews and inspections have been proven to provide the most cost effective method for finding problems early in the development cycle. Many times, I have had to bite my tongue to avoid blurting out to project managers, "The requirements are absolute garbage and there's no way that we can begin a productive testing effort until you clean up your act." Actually, this phrase would contain unprintable language and be uttered under one's breath. We have undoubtedly all shared this fantasy, and the ugly truth is that despite this valid complaint, the product delivery deadline is fast approaching.

Although I do not advocate cutting corners, there are some shortcuts that will help document the testing activities. A crude list of tests is better than no list. The minimum you will have is a documented trail, though rudimentary, that records your testing effort should you need to prove or demonstrate what you did."

—

"Just knowing how to get started with testing is a feat in itself. The tester must understand how to transform product information into test cases; this is the book's chief goal. Many existing books do an outstanding job of explaining software testing concepts and methods. Rather than reiterate what others have written, I make many references to their work. This book is a primer on getting started. It supplements currently available literature on software testing by providing an introduction to known software testing techniques."

—

"Testing is a responsibility shared with the rest of the development team. The old view of testing as an afterthought — design, code, and then you test — has never produced good testing results. The adversarial and destructive "developer vs. tester" mentality has often resulted from the developers' ignorance about software testing activities — more proof that testing is a unique discipline. It is often the case that a tester often knows more about programming than a developer knows about software testing. A collaborative approach between testers and developers fosters goodwill and good communication. By working closely with the testers, many developers learn more about software testing, even if all the developers see is how their knowledge about the product filters into the test documentation. Effective software testing requires cooperation among all the members of a project."

Topics and Outline

1. Tackling the Testing Maze.
Introduction.
Sample Application.
The Incremental Testing Approach.
Next Steps.
Summary.

(Continued on page 6)

Book Review—Introducing Software Testing

(Continued from page 5)

2. Test Outlines.
Introduction.
Sample Application.
Extracting the Requirements.
The Outline Approach.
Evaluating the Outline.
Schedule Estimation.
Summary.

3. From Test Outline to Test Cases.
Introduction.
Creating Test Cases.
Documentation Shortcuts.
Summary.

4. Using Tables and Spreadsheets.
Introduction.
Sample Application.
Documenting Test Cases.
Summary.

5. Other Types of Tables.
Introduction.
State Machines.
Test Case Table with Multiple Inputs.
Decision Tables.
Applications with Complex Data.
Managing Tests.
Summary.

6. Testing Object-oriented Software.
Introduction.
Comparing Object-oriented and Procedural Software.
System Testing Example.
Unit Testing of Classes.
Summary.

7. Testing Web Applications.
Introduction.
Sample Application.
Functional and Usability Issues.
Configuration and Compatibility Testing.
Reliability and Availability.
Performance.
Security Testing.
End-to-end Transaction Testing.
Database Testing.
Post-implementation Testing.
Summary.

8. Reducing the Number of Test Cases.
Introduction.
Prioritization Guidelines.
Priority Category Scheme.
Risk Analysis.
Interview ing to Identify Problem Areas.
Combination Schemes.
Tracking Selected Test Cases.
Summary.

9. Creating Quality Software.
Introduction.
Development Environmental Infrastructure.
Requirements.
Project Management.
Software Configuration Management.
Software Quality Assurance.
Reviews and Inspections.
Software Testing Environment.
Unit Testing.
Integration Testing.
System Testing.
Regression Testing.
Acceptance Testing.
Software Testing Tasks.
Test Planning.
Test Automation.
Problem Reporting System.
Test Reporting.
Summary.

10. Applying Software Standards to Test Documentation.
Introduction.
Common Elements.
Configuration Management.
Reviews.
Requirements Traceability.
Industry Standards.
Complying with the Standards.
Summary.

Good Companion Books

Software Testing by Ron Patton

Testing Computer Software by Cem Kaner, Hung Quoc Nguyen and Peter Falk

Surviving the Top Ten Challenges of Software Testing by William E. Perry and Randall W. Rice

Summary

I can highly recommend this book to anyone in testing, whether you are the poor soul that has been assigned to test your company's latest project and know nothing about testing, or if you are a test manager looking for a resource to build your team's skills. This book covers the major topics in testing in a way that promotes process-driven and requirements-based project structures, but the techniques can be used in any project environment. This is a must-have book for your technical bookshelf.

Other Information About the Book

To see a complete table of contents, visit http://www.aw.com/catalog/academic/product/1,4096,0201719746,00.html?type=TOC

To read the complete preface, visit http://www.aw.com/catalog/academic/product/1,4096,0201719746,00.html?type=PRE

Reviewed by Randall W. Rice

Q&A

(Continued from page 5)

. Constants and view definition tests (for explorer applications) and also any modified areas of the system against the original business case, business requirements, detailed design report (from the developers). Acceptance tests are end user specific but what would you say should be the essential tests for an acceptance test?

A: You have described exactly my viewpoint on User Acceptance Testing (UAT). Another this that is often seen in UAT are acceptance criteria, which is usually a list of system objectives from the user standpoint. In most of the cases I have seen in UAT, the act of "acceptance" is a foregone conclusion. The UAT in those cases is used to find gaps between the business processes and the system processes.

When people base UAT on requirements and specs, they run the risk of testing against the "paper world." Of course, requirements can contain errors and omissions. This kind of requirement-based testing is verification. Testing against real-world conditions is validation, which seems to be less understood as time goes by. Testing against requirements and use cases is great, but validation still needs to be performed against real-world scenarios.

Kinds of testing I have found effective during UAT include usability, integration, interoperability, and business scenario testing. I often tell people that the greatest value a user acceptance tester has is to bring the big picture of business processes to the test. If a user is spending a lot of time testing user interfaces, edits, errors, etc., they are probably working at a too detailed level. Although a business scenario test will also cover errors, edits, etc., I advise the process to be the focus, not the edits and other minor test cases.

Have you checked out my online demo for UAT training? If not, you can access it at:

http://www.riceconsulting.com/training/web-based/online_uat.htm

"I can highly recommend this book to anyone in testing, whether you are the poor soul that has been assigned to test your company's latest project and know nothing about testing, or if you are a test manager looking for a resource to build your team's skills."

The Software Quality Advisor Online

August, 2002

P.O. Box 891284
Oklahoma City, OK 73189

405-793-7449
405-793-7454 Fax
rrice@riceconsulting.com

"Test everything. Hold onto the good."
I Thessalonians 5:21

We're on the Web!
www.riceconsulting.com

September 2002 Issue:

Coming to Madison!

RCS Consulting Excellence!

October 3—4, 2002

A Two-day course in Becoming an Effective Test Team Leader

Presented by the Wisconsin QA Association.
(www.wisqa.org), and
Rice Consulting Services, LLC
(www.riceconsulting.com),
a world recognized leader in Quality and Testing Training.

Calendar of Events

Practical Software Quality Techniques (PSQT) Conference — Minneapolis, MN

Featured Session—Testing Dirty Systems (September 10, 2002)

Track Session—Surviving the Top Ten Challenges of Software Testing (September 11, 2002)

www.softdim.com

How to Become an Effective Test Team Leader, Madison, WI, October 3 — 4, 2002

Sponsored by the Wisconsin QA Chapter.

Register at www.riceconsulting.com/madison2002.htm

A Three-day Course in Web Testing

Chicago, IL, October 30—November 1, 2002

Sponsored by the Process Management Group, Ltd. And Rice Consulting Solutions, LLC

www.riceconsulting.com

EUROStar Conference

November 11—15, 2002

Edinburgh , Scotland

Randy will be presenting a one-day tutorial on *Surviving the Top Ten Challenges of Software Testing* and a keynote address on *Getting the Most Value from Every Person on Your Test Team.*

We hope to see you at one of these events!

If you have a group of 12 or more people in your city that would like to sponsor a training event, contact Randy Rice at rrice@riceconsulting.com to find out how to book a special presentation.