The Software Quality Advisor Online

Rice Consulting Services, Inc.
P.O. Box 891284
Oklahoma City, OK 73189
405-793-7449
405-793-7454 FAX
September 2001 Newsletter

The Role of Regression Testing in Dirty Systems

by Randy Rice, CQA, CSTE

This article is part of a work in progress, Testing Dirty Systems, co-authored with William E. Perry. A dirty system is one that is undocumented, patched over by constant maintenance, and unstructured. This information is also taken from our new training course, Integration and Interoperability Testing.

A type of test case that needs to be present in testing dirty systems is the regression test case. It seems that dirty systems are especially exposed to the regression risk, which is seen when a change to a validated piece of software causes a new defect to occur. Regression defects are a fact of life in software, especially software maintenance, which forces people to test more than just changes.

The key issue in regression testing is knowing how many test cases are needed to test a new release. The answer to this issue depends on:

The relative risk of the system being tested

If the potential impact of defects are minimal, then to test a large number of cases every time a change is made would be overkill. However, if property or safety are at risk, a large number of regression test cases would be very appropriate.

The level of system integration

This is a two-edged sword, in that on one hand highly integrated systems seem to be prone to regression defects due to the complex nature of many interfaces. A change in one module could manifest a defect in another module far downstream in the processing flow. On the other hand, highly integrated systems are difficult to regression test because of the large number of possible test cases required to adequately cover the integration paths. If we could predict where the defects might be, we wouldn't need to perform regression testing. However, that's not the case with most dirty systems.

The scope of the change

This is also a difficult criteria to define exactly. It is tempting to want to reduce the level of regression testing because a change might be very small. However, experience tells us that some major software failures can be traced back to a singe simple change. Consider the following examples:

On January 15, 1990, 114 switching computers in the AT&T telephone network crashed because of a single coding error. A misplaced "break" command in a C language program caused local computers to go down and broadcast "out of service" messages to be broadcast. The condition lasted for over nine hours in which switches failed, rebooted, only to fail again when restarted. The estimated cost of the outage was $60 million dollars and loss of company reputation. (Reference Globe and Mail, November 12, 1990, Page B8)

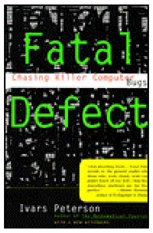
A spacecraft headed toward Venus, the Atlas-Agena, was blown up after it became unstable at an altitude of 90 miles. The problem was traced back to a missing hyphen in the flight plan. The cost of the spacecraft was $18 million. (Computer-Related Risks by Peter G. Neumann, Pg 26)

Of course, these are very visible examples of notable failures and thankfully, these don't happen to this extent every day. The point we are making by presenting them is that a small defect can have a huge impact. You can find a listing of many other classics software problems at: http://www.softwareqatest.com/qatfaq1.html#FAQ1_3.

Boris Beizer has also been quoted as saying that the top five software problems in his list are all related to unit defects. They are:

1. The Voyager bug (sent the probe into the sun).
2. The AT&T bug that took out 1/3 of US telephones.
3. The DCS bug that took out another 1/3 a few months later.
4. The Intel Pentium chip bug (it was software, not hardware).
5. The Ariane V bug.[1]

In a follow-up letter, Beizer stated that the Therac 25 defect which was responsible for the maiming and deaths of six people by overdosing of radiation therapy was notorious, but not a unit defect. You can find a complete description of that defect in the book, *Fatal Defect* by Ivars Peterson.



The resources available to perform regression testing

These resources include time, environments, people and tools. There are times when you can see the need to perform a certain level of regression testing, but are constrained by the lack of resources. This is a real-world situation which goes back to management support of testing. People can only do the job they have the resources to perform. Regression testing without automated test tools is so imprecise and laborious it could well be called "pseudo-regression testing."

A Risk-Based Process for Regression Testing of Dirty Systems

The best advice I can give for the regression testing of dirty systems is to base the extent of testing on relative risk. If the risk is high, you will want to develop a repeatable set of test cases that represents the widest scope of testing possible and perform those tests each time a change is made to the software. If the risk is low, you can test with a subset of regression test cases.

The following diagrams illustrate the effect of segmenting regression test cases by risk. In each diagram a universe of test cases is defined. However, we are quick to agree that number of possible test cases approaches infinity for many applications. The universe as shown in the diagrams is a practical "line-of-sight" view of those test cases known to be needed and effective.

In Figure 1, the high-risk environment contains all test cases that are known to exist.

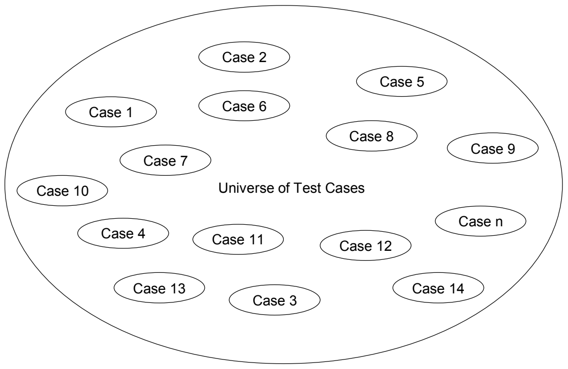


Figure 1 – The High Risk Test Case Universe

In Figure 2, the test cases are segmented by risk. In this environment, there are some parts of the application that can be regression tested at lesser degrees than others.

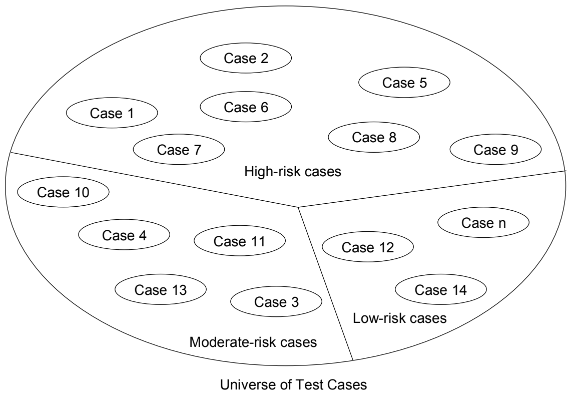


Figure 2 – Multiple Levels of Risk in the Universe of Test Cases

[1] Letter to swtest-discuss, 10 June 1997

We must be quick to point out that in dirty systems, and even those are clean, that varying levels of risk can be seen spread across a function. When this occurs, the multiple test cases that may be required to test such a function will also have varying levels of risk. We mention this because sometimes in testing the high-risk functions you must also test the low risk functions.

In Figure 3 the effect of varying levels of risk in a transaction or major function are shown by the linking together of test cases that are required to test the major function.

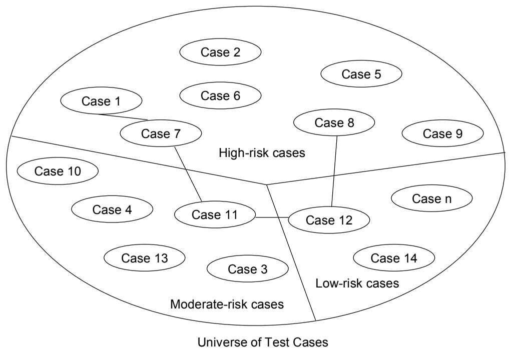


Figure 3 - Varying Levels of Risk in a Major Function or Transaction

This is perhaps the most difficult of the regression test situations because the risk cuts across all levels. In such a situation, the regression test would be required to include all of the test cases, regardless of the lower levels of risk seen in some of the cases.

Common Regression Test Approaches

The following regression test approaches are not necessarily right or wrong for you. The approach depends on your risk and the application you are testing. This is simply a list of some of the more common ways regression testing is applied.

- **Test Everything, Every Time**
  Rationale: The risk of failure in a particular application is so high that it outweighs the cost of massive test efforts.
  Benefits: A high level of test coverage is achieved.
  Risks: If done manually, the chances are low that every case is actually being tested, since there is a high likelihood of human error. In addition, the burden of performing such an intense test manually can lead to tester burnout.

- **Test a Few Cases Every Time a Change is Made**
  Rationale: The risk of failure is too low or the deadlines are too close to justify testing a large number of cases.
  Benefits: If the risk is actually low, the level of testing matches the level of risk.
  Risks: If the risk is higher than estimated, the level of regression testing may be too low.

- **Test Even When a Change is Not Made**
  Rationale: The risk of failure is high and there are external factors that are not under the control of software configuration management that could impact systematic behavior.
  Benefits: The application is undergoing continuous monitoring.
  Risks: Unless automated, this level of testing can be overwhelming. Sometimes a false sense of security can be realized since the regression test is only as complete as the defined test cases.

- **Test Critical Cases Every Time a Change is Made**

  Rationale: The risk of failure is low to moderate and there is a way to assess relative risk. Testing is optimized to increase value and decrease redundancy. This method is seen many times when test automation is not used, or when the possible number of regression test cases is large, even for the use of automated test tools.
  Benefits: Testing matches risk. For manual testing, there is a way to balance the need for regression testing with the realization that there will always be a risk of missing the definition or performance of a test case.
  Risks: A test case that is necessary may not be included in the regression set.

- **Test Critical Cases Even When a Change is Not Made**

  Rationale: The risk of failure is low to moderate and there are external factors that are not under the control of software configuration management that could impact systematic behavior.
  Benefits: The application is undergoing continuous monitoring.
  Risks: Unless automated, this level of testing can be overwhelming. Sometimes a false sense of security can be realized since the regression test is only as complete as the defined test cases. A test case that is necessary may not be included in the regression set.

Regression Testing Approach Checklist

#	Criteria	Yes	No
1	Should your application fail, would the negative impact be high?		
2	Are changes being made to any part of the system on a frequent basis?		
3	Are changes being made to any part of the system outside of the control of your organization?		
4	Is it possible for you to define a set of regression test cases that completely defines all functionality?		
5	Is it possible for you to define a set of regression test cases that completely defines critical functionality?		

The above checklist is subjective in points such as "high impact" and "on a frequent basis." This subjectivity is intentional, since impact and frequency are relative to each application environment. Therefore, you need what constitutes "high impact" and "frequent basis" for your organization and applications.

The purpose of the checklist is to lead you through the questions that will help you determine the level of regression testing that is most appropriate for you. For example, if you answer questions 1, 3 and 4 with a "yes", then you could be a candidate for testing every condition on a regular interval.

The Regression Testing Process

In the above process, the following steps are performed:

Step 1 - Test existing software using test data containing pre-modified test cases. The results of this test will be the baseline to compare against.

Step 2 – The software/system is modified.

Step 3 - Modify the test data to contain new test cases to validate changes.

Step 4 - Test modified software using modified test data.

Step 5 - Compare the post-modified test results with the pre-modified test results. Any differences should be identified as potential defects.

Continually Building the Regression Set

As the application continues to undergo maintenance, new regression test cases will need to be added. These cases will be derived from cases required to test new functionality, and from test cases created from the identification and repair of defects (Figure 5).

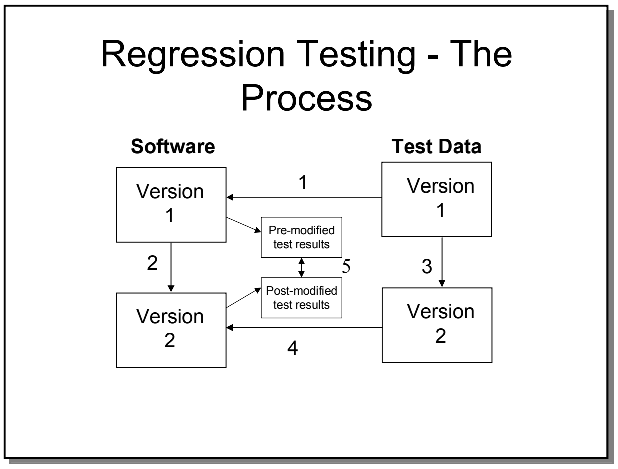

Regression Testing - The Process
Software
Test Data
Version 1
Version 1
1
Pre-modified test results
2
5
3
Post-modified test results
Version 2
Version 2
4

Figure 4 – The Regression Testing Process

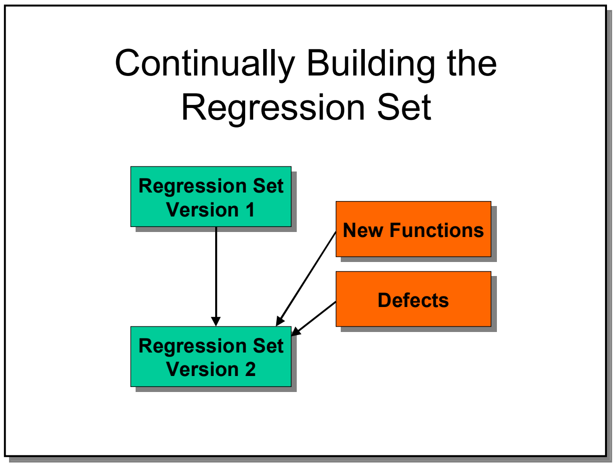


Figure 5 – Continually Building the Regression Set

Tips for Performing Regression Testing

- **Control the scope of testing.**

You only have so much time for testing, so choose your tests carefully.

- **Build a reusable test bed of data.**

A reusable test bed of data is essential for regression testing.

- **Use automated tools.**

Especially when it comes to on-line regression testing, automated capture/playback tools are the only way to achieve exact regression testing.

- **Base the amount of regression testing on risk.**

By its very nature, regression testing is redundant. You can manage the redundancy by basing your testing on risk.

- **Build a repeatable and defined process for regression testing.**

This adds rigor and consistency to the regression test.

Summary

In dealing with all of the uncertainty, complexity and volume of regression test cases, one of the best things you can do to make regression testing manageable is to control the scope of it by carefully defining the essential cases. Automated test tools can be effective vehicles for reducing the manual testing burden, but there is still the need to design test cases. In fact, regression testing is the perfect application for automated test tools, as they can perform identical test actions multiple times and compare test results with exact precision. However, regression testing in highly integrated and unstructured applications can be very difficult to perform and maintain, even with automated testing tools. Hopefully, using some of the concepts and techniques presented in this article, you can plan and perform regression testing in a way that is both effective and efficient.

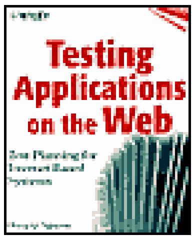
Book Review

by Randy Rice, CQA, CSTE

Testing Applications on the Web

★★★★½

By Hung Q. Nguyen



Format: Paperback, 416pp.
ISBN: 047139470X
Publisher: Wiley, John & Sons, Incorporated
Pub. Date: October 2000

This is a good book on web testing with plenty of details and examples. When I first read the book, I wished there would have been more emphasis on process. Although web-based projects are not known for their use of processes, it would be nice to start the discussion someplace. However, after reading the other book reviewed this month, *Automated Web Toolkit*, I realized that these two books compliment each other very well.

Testing Applications on the Web covers the essentials of web testing and what makes testing web-based applications different from other technologies. This book lays a foundation of understanding basic testing methodology, understanding networking essentials, web components, and test planning. Then, specific test practices and techniques are discussed, including user interface tests, functional tests, database tests, help tests, installation tests, compatibility tests, load testing, and web security concerns. Web test tools are also discussed at an overview level.

What I Liked About This Book

I liked the specific examples, especially in test case design and load testing. I also liked that there is a "Why Read This Chapter?" at the beginning of each chapter.

What I Would Have Liked to Have Seen

If this book had a unified process to wrap around all of the great techniques, I would have rated it as 5 stars.

Scoring

Readability - 5
Breadth of coverage – 5
Depth of discussion - 5
Accuracy - 5
Credibility - 5
Organization - 3
Overall Score – 4.6

Topics

Welcome to Web Testing.
Web Testing versus Traditional Testing.
METHODOLOGY AND TECHNOLOGY.
Software Testing Basics.
Networking Basics.
Web Application Components.
Test Planning Fundamentals.
Sample Application.
Sample Test Plan.
TESTING PRACTICES.
User Interface Tests.
Functional Tests.
Database Tests.
Help Tests.
Installation Tests.
Configuration and Compatibility Tests.
Web Security Concerns.
Performance, Load, and Stress Tests.
Web Testing Tools.
Finding Additional Information.
Appendices.
Index.

Summary

This book has a lot of good web-based testing techniques and specific examples. It belongs on every web tester's bookshelf.

Reviewer Randy Rice, CQA, CSTE

Automated Web Testing Toolkit

By Diana Stottlemyer

Paperback - 285 pages (2001)
John Wiley & Sons; ISBN: 0471414352

Writing a book on any type of test automation is certainly a daunting project, as the topic is constantly changing, both in terms of the tools and vendors, and in terms of the web itself. This book addresses the framework of software testing on a web project, project management for web projects, and then the support that can be accomplished with web-based testing tools.

This book would be a start for people just getting into testing a web site, but those who have been down the web testing road for awhile may be disappointed. Actually, I thought this book was a good compliment to Hung Nguyen's book, *Testing Applications on the Web*. The reason I make this comment is that Nguyen's book has a lot of concrete examples of web-based test cases but seemed short on the overall process to manage testing web applications. *Automated Web Testing Toolkit* is long on the process side, but seemed short on the practical examples of how to apply the test tools in specific test cases.

What I Liked About This Book

To me, the strong point of this book was the emphasis on processes and sound project management – things we at RCS seldom see done well on web projects. I also liked the discussion of risk assessment and its importance to testing, but I really would have liked to have seen an example of how to quantify risks on a web-based project. I found the chapters on testing different platforms and servers, and on load testing very helpful. I did pick up on some tools I was not aware of previously and, finally, the templates and test plan examples were a nice inclusion.

What I Would Have Liked to Have Seen

In a word – details. The chapter on web test tools is less than 30 pages long and could really use more screen shots and trade-off information about the various tools within a category. One big gap was the short treatment of web-based capture playback tools. In the discussion of load testing, there was a lot of emphasis on testing the web server, but little discussion on testing those transactions all the way into back-end processes.

Scoring

Readability - 5
Breadth of coverage – 3
Depth of discussion - 2
Accuracy - 4
Credibility - 4
Organization - 4
Overall Score – 3.6

Contents

Pt. 1	Managing the Web Testing Process
Ch. 1	The Web Testing Process
Ch. 2	Testing Methodology
Ch. 3	Web Site Management
Ch. 4	Risk Management
Pt. 2	Web Testing Tools and Techniques
Ch. 5	Web Site Testing Tools
Ch. 6	Preparing the Web Environment for Testing
Ch. 7	Testing Languages and Databases
Ch. 8	Testing on Different Platforms and Servers

Ch. 9	Web Capacity Testing-Load and Stress
Ch. 10	Running the Web Test
Ch. 11	Analyzing the Test Process and Documentation
Pt. 3	Templates

Summary

People on web projects need a process and tools to test effectively. This book is quick read and a high-level start, but be prepared to do further research on the "how to" questions.

Reviewer Randy Rice, CQA, CSTE

Frequently Asked Questions

by Randy Rice, CQA, CSTE

Q: I am greatly impressed to see your site. I want to get information about CMM levels and standards.

A: The place to go to find out this information is the Software Engineering's Web Site at: http://www.sei.cmu.edu/cmm/cmms/cmms.html

You can also find some very good articles on the CMM at
http://www.stsc.hill.af.mil/crosstalk/

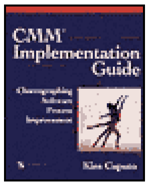
Select "Search Crosstalk Issues", then search on CMM and you will find many hits. One in particular that you may like is called The Capability Maturity Model: A Summary - May 99 the URL is http://www.stsc.hill.af.mil/crosstalk/1999/may/paulk.asp.

Q: I want a thing like complete manual of CMM standards. Will you please tell me how to get it?

A: The CMM is not a standard, but an assessment framework. A company can build software anyway they like as long as the processes are defined, repeatable, etc. You get a CMM designation based on how you meet the criteria for process maturity, people maturity, etc. The ISO standards are very similar. As they apply to software, the ISO 9000-3 standard for software is fairly short and contains just basic standards that you would find in any testing book. Therefore, people have developed other certifications, such as SPICE (http://www-sqi.cit.gu.edu.au/spice/).

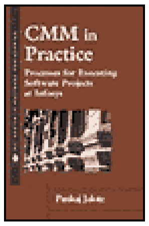
You can also find many other links at http://www.tantara.ab.ca/info.htm

Here's where you can find more complete information about the CMM:



CMM Implementation Guide: Choreographing Software Process Improvement (Book/CD) by Kim Caputo

This book describes the CMM and how to implement it in an organization.



CMM in Practice: Processes for Executing Software Projects at Infosys by Pankaj Jalote

This book shows how one company built a project at Level 4 of the CMM - Lots of templates and examples.

Comparing ISO 9000, Malcolm Balbridge, and the SEI CMM for Software: A Reference and Selection Guide by Michael O. Tingey

This book compares these approaches to software quality.

You can get these books from bn.com or amazon.com.

Rice Consulting Services, Inc.
P.O. Box 891284
Oklahoma City, OK 73189
405-793-7449
405-793-7454 FAX

Coming to Chicago!
October 30 – November 1, 2001

Q: How does a manager determine how long User Acceptance Testing (UAT) should take? Is it possible to base it on the number of use cases or business requirements that have been written? How do I determine how manual users should be involved with UAT? Also, are users responsible for writing UAT scripts to test the software?

A: As I tell people in our testing courses, when it comes to measurement and estimating, actual mileage varies because of differences in technology, organization, processes, etc.

However, with that caveat, there are some guidelines that seem to hold on an average.

First, you can estimate about one-third to one-half of the project time going to test-related activities. Of that amount, we normally see about 10% of testing allocated to UAT if adequate unit and system testing has preceded it. Otherwise, UAT can take 25% of the testing time budget.

Yes, it is good practice to base the estimate in use cases or business requirements to be tested. For use cases, you need to determine the number of process variations. Each of these will be the basis of a test script. My estimating guideline is that it takes about 5 times as long to test a process than to simply perform it. This is due to extra time required to observe and document results, report defects, perhaps repeat the test to confirm a defect and to re-test the function once a defect is fixed. Keep in mind that the 5 times multiplier is just an average.

Business requirements are a bit trickier, because you must decompose them into sub-functions first, and then test cases. This is basically the same analysis that happens in developing the use case. Once the sub-functions have been identified, the same 5x multiplier can be applied.

Planning time can equal or exceed execution time. In fact, another guideline I often use is one-third of test time for test planning, one-third for test execution, and one-third for test evaluation and reporting. This can be seen by test phase for by testing as a whole.

How many users to have involved depends on the size of the project and how many people the test leader can reasonably manage. In my experience, a team of 12 full-time user testers to test an enterprise system is about the most one leader can keep track of. If the system size justifies more resources, then you will want to divide and conquer by forming multiple teams. You will need to look at skill levels both with the business and with the technology to estimate the number of testers. If you have a bunch of people that are struggling with the system, progress will be slow. One way to help judge this is by having one or two users as part of a system test team. Another way is to sample some of the users to have them perform some of the test scripts for about an hour before the UAT test team is formed.

My personal opinion is that the users should own the UAT. That means they are responsible for creating the high-level test plan, the UAT acceptance criteria, designing and writing the tests to be performed, perhaps creating test data, and evaluating test results. Most UAT teams need assistance from developers and testers to keep the process on track, and that's fine, as long as the users keep ownership of the test. Users must be prepared to hold the line, if necessary, when they see functions that simply do not support the business. A classic situation is where the development team tries to sell the users on workarounds instead of fixing the process.

One final word about UAT is that you can leverage a lot of the effort and save a lot of grief by involving users early in the development or purchase process in reviews, walkthroughs, etc. I know this is easier said than done, but I always tell people that UAT is one of the most necessary kinds of testing, done at the worst possible time.

Q: Can you help me out on the testing front? We are now in bug fixing mode for a client server application. What should my testing philosophy be?

A: This is a very involved question, but here is my short answer.

I know you are past this stage, but testing starts as the software is being written. Each developer needs to test his/her own code functionally (externally) and structurally (internally). Then, as the entire application starts to come together you can test it as a system.

In fix and debug mode, you need to pay attention to two things:

1) that the changes work correctly (functional tests), and

2) that the changes do not break things that used to work (regression tests).

This requires a set of test cases that includes tests for the bug fixes and tests that should be performed each time the software is changed. If you are getting a new build everyday, this can be overwhelming. Test tools can help, but only to the extent that you know what to test and have a good process in place for testing.

Metrics and Measurements

by Carl Chandler

Metrics and measurements are very important in the maturity of your process because they will allow you to monitor your progress and continually improve. To look at it in another way, process maturity is much like taking a trip. You must know where you are now, where you want to be, have a plan for how to get there, and a way to measure your progress.

That is where metrics and measurements come in. First, lets take a look at terminology:

Measurement – The extent, dimensions, capacity, etc. of anything, esp. as determined by a standard.

Examples: Mileage, Function points, Number of Defects

Metrics – A measurement per another measurement.

Examples: Miles per hour, Defects per function point

Two valuable measurements are time, and defects.

Time
- for future estimating

Defects
- for determining effectiveness of testing
- for improving the development and testing processes

Combined they are valuable test metrics

Some examples of simple, yet effective metrics from our course ***Becoming an Effective Test Team Leader*** include:

Time:
- Time per test case
- Time per test script
- Time per unit test
- Time per system test

Sizing:
- Function points
- Lines of code

Defects:
- Numbers of defects
- Defects per sizing measure
- Defects per phase of testing
- Defect origin
- Defect removal efficiency

We teach in our course that the most valuable test metrics you can capture and track is defect removal efficiency. This metric tells how efficient your testing process is by dividing the number of defects you find vs. the number of defects found in the product during its defined operational life.

Defect Removal Efficiency =

$$\frac{\text{Number of defects found in producer testing}}{\text{Number of defects during the life of the product}}$$

For example, if you find 90 defects during inspections, unit testing, system testing, and user acceptance testing – then your customer finds 10 more defects in the next year, your defects removal efficiency is 90%.

Rice Consulting Services' Consulting Offerings:

Testing Assessments

Rice Consulting Services' testing assessment is a quick and effective way for an organization to determine where they are in terms of software testing maturity. The assessment looks at three areas that are critical to testing:

• **Test organization** - Who performs testing, what levels of experience are present, and when testing is performed in the development/maintenance life cycle.

• **Test process maturity** - How well-defined, well-deployed, and repeatable the test process is, and whether it incorporates good testing management, practices, tools, and techniques.

• **Readiness** - An assessment of the organization's readiness to improve the testing process. This involves an assessment of the staff's testing awareness, testing skills, and motivation to change current practices.
The deliverable is a report detailing the assessment's findings, a recommended quality improvement strategy, and a plan for addressing the improvement needs identified. If the assessment uncovers the need for in-house skills training and consulting, we will include proposed training and consulting plans in the report. The report is typically about 15 pages in length.

Rice Consulting Services' Course Offerings:

If you would like to learn more about the information covered in Carl's article we at Rice Consulting Services, Inc. offer an excellent course that will enhance your company's software quality process.

Building an Effective QA and Testing Process for Ongoing Validation

– 2 days

This course is designed to teach participants how to design and implement processes for quality assurance and quality control. The benefit from these processes is the continued quality of existing systems during ongoing operations and maintenance.

This session is appropriate for software developers and managers, quality assurance and testing personnel, and systems support professionals.

The workshop contains two team-based exercises which focus on having the participants write processes for ongoing testing and change control using their own organization's unique attributes.

Interoperability and Integration Testing

– 2 days, Intermediate

This course is designed to present strategies and techniques for testing within a framework of diverse technologies and applications. It is assumed that the attendees will have a working knowledge of testing and test terminology. Attendees will learn how to plan, conduct and evaluate tests in diverse technology environments, especially when the applications in those environments interact together. The testing of Commercial Off-the-shelf Software (COTS) will be discussed, along with the role of regression testing, configuration management, automated test tools and ongoing validation in diverse technical environments. Attendees will leave this course with a solid foundation for testing in situations which are very diverse and dynamic.
Attendees will learn

- The basic issues and risk of integration and interoperability testing
- The deeper issues of performing a risk assessment
- Processes for integration and interoperability testing and configuration management in diverse environments
- How to leverage test tools in diverse environments
- The process for performing regression testing in diverse environments
- How to build and manage a test environment that starts to simulate the operational environment
- How to measure Return on Investment (ROI) in a Commercial Off-the-shelf Software (COTS) environment

- The impact of various lifecycle models on integration and interoperability testing
- How to keep an application in a diverse environment in a validated state

Becoming an Effective Test Team Leader

2 days

This two-day session is designed for test leaders and test managers, people who expect to be in a test leadership role, or people who lead other test managers and test leaders. The main objective of this session is to teach you how to be the very best test manager and leader. This course also answers the question, "What does it mean to be the best?" There are many people functioning as test managers, but how many are really leading the team? In leading a test team, you must not only understand the basics of software testing, but you must also understand your own organizational culture. Once you understand your organizational culture, you might find that testers have a less than positive image. This session will discuss how to transform the image of testers from one of police to one of team members.

You will learn the terminology, process, and challenges of testing in the real world. Team-based exercises reinforce the concepts of facilitating team activities and performing leadership activities.

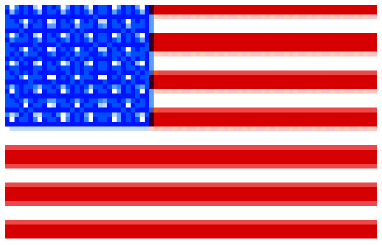
As a result of attending this seminar, you should have a good working knowledge of software testing and what it takes to design and conduct an effective test of software, regardless of the technology.

Becoming an Effective Test Team Leader will help you become more comfortable and confident in leading the testing effort in your organization. You will emerge from this two-day session knowing how to develop test cases and test plans. You will also leave with a knowledge of how tools can help you perform testing.

Sometimes people feel intimidated by the technical aspects of software testing and lack the confidence they need to be credible test leaders in their organization. Learn the issues and processes for effectively testing software by attending this hands-on course.

For more information on this course or one of our many other offerings please contact Carl Chandler at (405) 414-6759

**Rice Consulting Services, Inc.
P.O. Box 891284
Oklahoma City, OK 73189
405-793-7449
405-793-7454 FAX**



To our friends in New York City, Washington D.C. and Pennsylvania - You stood by us in Oklahoma City during our time of need and we stand by you at this time. We are praying for you daily. God bless America.

Note from Randy –

This is a time where we as Americans are faced with some great challenges. I received this letter by one of our state representatives in e-mail the other day. I don't know this gentleman's political affiliation, but when I read it, I said "Amen." We at RCS are grieving with those who have lost loved ones in this attack on America. We also have all too clear in our memories when we had to deal with the grief of 168 of our friends and neighbors in the bombing of the Federal Building in Oklahoma City. The thing that kept us going then and will keep us going in this time is to tighten our bootstraps and manage to do our jobs. All of this is not meant to be a political statement, but rather an encouragement to stay strong. One of our main missions at RCS is to help you do your job better. So, with these thoughts, I respectfully share this letter with you from Dave Herbert.

Randy Rice, 9/19/2001

America needs to toughen up, and do it now. This is going to be a long drawn out action against these terrorists. The likelihood of similar terrorist attacks is greater now than ever before. We must quit cowering in our corners, and wringing our hands. We need to suck it up, and make sure we don't kill our own economy by hiding and hording. Break out the flags, stick out your chest, and show the American spirit can't be broken. This will not be a war of sacrifice like WWII. We don't need to save our metal cans and conserve on spending. This is an economic war. The more we conserve the more they win. The idea of the terrorists is to bring down the American economy, thereby throwing the world economy into a tailspin. We need to go about our business as usual, and show these terrorists our true colors. Roosevelt was right when he said. "The only thing we have to fear is fear itself". In the midst of all this death and destruction, we should remember the strength of the British during the bombings by the nazi's in the Second World War. This terrorist war is a war that has been going on for years, and we have looked the other way for far too long. We should never forget the lost Americans, and the grief of the families, but we cannot afford to let them see us sweat.

The best thing we can do is to continue to fuel our economy. Buy something. Keep the cash flowing in this country. An economic war is the toughest kind of war, because it goes against all theories of wartime. These terrorists know fear stops spending. Use your tax refund to purchase something, or send it back to the president, and tell him to use in the war effort. If you want to bring these terrorists to their knees, you must increase your productivity, and do it with a smile on your face knowing you are striking a blow for America. The terrible sacrifice of human lives should not be forgotten, but remembered everyday, and honored with our sweat, and productivity. Whatever your job is, it is essential to our economy, and at all costs we must keep our economy strong. Go to work for America. Work as if you were in a defense plant making weapons to fight our enemy. If you are a car salesperson, sell more cars. If you are a ditch digger, dig better and faster. If you sell clothing, turn on the charm and sell more. If your job has a quota, raise it. If you are a construction worker, work harder and faster. The fuel we need to win this war is not jet fuel, but economic fuel. It is vital that we heat up our economy. Wear a flag pin, and fly Old Glory from your home and office. Work for America, and the American way. The American worker is the best and most productive worker in the world, and now is the time to do what we do best. God Bless America!

Dave Herbert
Okla. State Senate #42

Links…

Expect – A Tool for Regression Testing Interactions

http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/expect/

Managing Object-Oriented Integration and Regression Testing – Without Being Drowned

http://www.informatik.fernuni-hagen.de/import/pi3/publikationen/abstracts/EuroSTAR98.pdf

Regression Testing article

http://sern.ucalgary.ca/~sdyck/courses/seng621/regression.html#Regression%20Procedure

A PowerPoint Slide Show on Software Testing – Great for self-study

http://sern.ucalgary.ca/~dsloane/testing3/sld001.htm

Mining System Tests to Aid Software Maintenance

http://xsuds.argreenhouse.com/papers/ieee.html

Notable Quotes…

Yesterday I was a dog. Today I'm a dog. Tomorrow I'll probably still be a dog. Sigh! There's so little hope for advancement.
- Charles M. Schulz, (Snoopy)

The difference between literature and journalism is that journalism is unreadable and literature is not read.
- Oscar Wilde (1854 - 1900)

Personally I'm always ready to learn, although I do not always like being taught.
- Sir Winston Churchill (1874 - 1965)

For lack of guidance a nation falls, but many advisers make victory sure.
- The Bible - Proverbs 11:14

**Rice Consulting Services, Inc.
P.O. Box 891284
Oklahoma City, OK 73189
405-793-7449
405-793-7454 FAX**

October 2001 Issue:

- **Foundations of User Acceptance Testing**
  by Randy Rice, CQA, CSTE
- **Walkthroughs, Reviews and Inspections**
  by Carl Chandler

Coming to Chicago!
October 30 – November 1, 2001

Rice Consulting Services, Inc., *a world recognized leader in Quality and Testing Training.*
is teaming with
Process Management Group, Ltd., *the Midwest's Premier Provider of Software Quality and Software Testing Services (www.pmgltd.com)*
To present to you a Three-day course in User-Oriented Practices for Delivering Quality Software
Log on to http://www.riceconsulting.com/chicago2001.htm *to learn more about the course and to register.*
Look for early-bird incentives!

This certificate worth _____ CPE credits* towards Certified Software Test Engineer Continuing Professional Education through the Quality Assurance Institute.

Solutions
QUALITY ASSURANCE INSTITUTE

*Category E - Self-Study Courses Activities designed to improve your proficiency in CSTE skill areas as defined in the *Common Body of Knowledge* may qualify for CPE credit up to a **maximum of 20 credits per year**. Qualifying activities include: Professional memberships that offer self-study education regarding quality assurance within information technology. It's not the membership that earns the credit, but the study materials provided by the membership.

To redeem complete the following information and submit to the Quality Assurance Institute at the time of reporting CPE credits.

Name: ______________________________

CSTE/CQA Certification Number: ______________________
(circle one)

Email Address: ______________________________

RCS
Consulting Excellence!

Credits available to members of The Software Quality Advisor only. To become a member of The Software Quality Advisor sign-up at http://www.riceconsulting.com/SQAdvisornew.htm

Rice Consulting Services, Inc.
P.O. Box 891284
Oklahoma City, OK 73189
(405) 793-7449 / (405) 793-7454 Fax
e-mail rcs@telepath.com
http://www.riceconsulting.com