Volume 5, Issue 5
May 2002—The Software Quality Advisor

The Software Quality Advisor Online

The Science of Software Testing

By Randall W. Rice, CSQA, CSTE

The Science of Software Testing

This article discusses software testing as a science, including:

- How scientific experiments are like tests
- How many points of terminology have similar meaning in both science and testing.
- How testing can apply scientific approaches to be performed at a high level of reliability.

Testing has been described as an art (*The Art of Software Testing* by Glenford Myers), a craft (*The Craft of Software Testing* by Brian Marrick), and a process (*Effective Methods for Software Testing* by William E. Perry), but I would like to examine another aspect of testing, that is, the Science of Software Testing.

A Brief Background

I graduated college with a Bachelor of Science degree as a Math major, which was accidental. I started out as an electrical engineering major but changed late in my Junior year when I discovered that EE really wasn't as appealing to me as I had originally thought it would be. In my schooling, I was trained in the traditional scientific method, which affects how I see the practice of science.

The Traditional Scientific Method

The traditional scientific method has been the predominant method for people to observe and understand the operation of world and the universe. In recent years, some scientists have developed methods that are less rigorous, but the traditional method is what I will use as the basis for this article. The steps in the traditional method are:

1. Observe some aspect of the universe.
2. Invent a theory that is consistent with what you have observed.
3. Use the theory to make predictions.
4. Test those predictions by experiments or further observations.
5. Modify the theory in the light of your results.
6. Go to step 3.

There are differing views among scientists today as to what constitutes a theory, a hypothesis and a fact. How someone defines these terms can greatly affect their view of science. To fully expound on

these differing views of the scientific method and how the terms are defined is beyond the scope of this article. It is important, however, to understand there is often a level of bias since people will hold certain definitions that are consistent with their beliefs of how the world operates. This is circular reasoning, because if I am trying to explain how and why something happens, it will be based in some degree of a framework that I believe in already. Therefore, one of the great challenges of science is to maintain objectivity.
(Continued on Page 2)

Inside this issue:

Book Review—*Peer Reviews in Software* by Karl Wiegers

I was delighted to read Karl Wiegers' latest work because of the need I see for a current resource for reviews that is easy to read and practical in application. Peer Reviews in Software meets both criteria – easy to read and full of practical ideas for implementing reviews in your organization.

The scope of peer reviews as covered in this book range from informal walkthroughs to formal inspections, although the most attention in the book is focused on inspections. Wiegers starts the book by laying a solid foundation of why review-based techniques are important. This information is an excellent resource for anyone wanting to make the case for early quality control in a project.

What I Liked About This Book

Wiegers describes in practical details the review process and how to apply it by using
(Continued on Page 2)

Book Review (Continued from Page 1)
Peer Reviews in Software by Karl Wiegers

practical examples. Most importantly, Wiegers never loses sight of the many human factors that will make or break the reviews effort in an organization. He provides plenty of practical advice on dealing with the challenges of implementing reviews.

Content

Topics covered in the book include:

- Overcoming resistance to reviews
- Inspection teams and roles
- Inspection process stages
- Scheduling inspection events
- Analyzing inspection data
- Peer review training
- Critical success factors and pitfalls
- Relating peer reviews to process improvement models

Scoring

Readability - 5

Breadth of coverage – 5

Depth of discussion - 5

Accuracy - 5

Credibility - 5

Organization - 5

Overall Score – 5

Summary

This is a very complete and easy to read book on the topic of software project reviews. The book's practicality should earn it a place on any software quality practitioner's or project manager's bookshelf.

Reviewer

Randy Rice, CSQA, CSTE

Format: Paperback, 256pp.
ISBN: 0201734850
Publisher: Addison Wesley Longman, Inc.
Pub. Date: December 2001

"Managers, developers, and customers sometimes oppose reviews because they believe reviews will cost too much and slow down the project. In reality, reviews don't slow the project—defects do."

Karl Wiegers

The Science of Software Testing (Continued from Page 1)

For the purpose of defining the working definitions of this article I will outline the following terms, which I do not propose to be perfect or accepted by everyone. The source is *Webster's Revised Unabridged Dictionary, © 1996, 1998 MICRA, Inc.*

Observation – "(a) The act of recognizing and noting some fact or occurrence in nature, as an aurora, a corona, or the structure of an animal. (b) Specifically, the act of measuring, with suitable instruments, some magnitude, as the time of an occultation, with a clock; the right ascension of a star, with a transit instrument and clock; the sun's altitude, or the distance of the moon from a star, with a sextant; the temperature, with a thermometer, etc. (c) The information so acquired.

Note: When a phenomenon is scrutinized as it occurs in nature, the act is termed an observation. When the conditions under which the phenomenon occurs are artificial, or arranged beforehand by the observer, the process is called an experiment. Experiment includes observation."

Experiment – "1. A trial or special observation, made to confirm or disprove something doubtful; esp., one under conditions determined by the experimenter; an act or operation undertaken in order to discover some unknown principle or effect, or to test, establish, or illustrate some suggest or known truth; practical test; poof."

Hypothesis – "1. A supposition; a proposition or principle which is supposed or taken for granted, in order to draw a conclusion or inference for proof of the point in question; something not proved, but assumed for the purpose of argument, or to account for a fact or an occurrence; as, the hypothesis that head winds detain an overdue steamer.

2. (Natural Science) A tentative theory or supposition provisionally adopted to explain certain facts, and to guide in the investigation of others; hence, frequently called a working hypothesis."

Assumption – "The thing supposed; a postulate, or proposition assumed; a supposition."

Theory – "1. A doctrine, or scheme of things, which terminates in speculation or contemplation, without a view to practice; hypothesis; speculation.

2. An exposition of the general or abstract principles of any science; as, the theory of music.

3. The science, as distinguished from the art; as, the theory and practice of medicine.

4. The philosophical explanation of phenomena, either physical or moral; as, Lavoisier's theory of combustion; Adam Smith's theory of moral sentiments."

Fact – "2. An effect produced or achieved; anything done or that comes to pass; an act; an event; a circumstance.

3. Reality; actuality; truth; as, he, in fact, excelled all the rest; the fact is, he was beaten.

(Continued on Page 3)

"...there is often a level of bias since people will hold certain definitions that are consistent with their beliefs of how the world operates. "

Keys to Successful User Acceptance Testing

By Randall W. Rice, CSQA, CSTE

Introduction

I get a lot of questions each month about user acceptance testing (UAT) – what it is, how to perform it, which (if any) tools to use, and a variety of other questions. The purpose of this article is to build on a base of past articles and continue to build a knowledge base and lessons learned. Who knows? One day all of these articles may turn into a book!

The keys presented in this article are not hidden from view, just often overlooked by people. If you ignore them, you will experience problems and less than effective user acceptance testing. If you heed them, you will open new doors to involving your users in testing.

User Acceptance Testing (UAT) Defined – User acceptance testing is the validation that a system or application will meet user needs in the operational or business environment.

This article can be applied from multiple perspectives:

- If you are a tester or test facilitator, this article is written directly to you. Your job is to help users plan and conduct an effective test to validate that the application will work correctly in their environment.
- If you are a user, focus the points in this article to you and other users that will be involved in the test.
- If you are a project manager, this article can provide valuable ways to involve users in the entire project, not just testing.
- If you are a developer, this article can help you understand what the users will need to do in testing the application. You can use the

(Continued on Page 6)

"User acceptance testing is the validation that a system or application will meet user needs in the operational or business environment."

The Science of Software Testing (Continued from Page 2)

4. The assertion or statement of a thing done or existing; sometimes, even when false, improperly put, by a transfer of meaning, for the thing done, or supposed to be done; a thing supposed or asserted to be done; as, history abounds with false facts."

Law – "5. In philosophy and physics: A rule of being, operation, or change, so certain and constant that it is conceived of as imposed by the will of God or by some controlling authority; as, the law of gravitation; the laws of motion; the law heredity; the laws of thought; the laws of cause and effect; law of self-preservation.

6. In mathematics: The rule according to which anything, as the change of value of a variable, or the value of the terms of a series, proceeds; mode or order of sequence.

7. In arts, works, games, etc.: The rules of construction, or of procedure, conforming to the conditions of success; a principle, maxim; or usage; as, the laws of poetry, of architecture, of courtesy, or of whist."

The Science of Software Testing

Some testing methods are performed at a "junk science" level, which are often based on small sample sizes and poorly controlled or documented experiments. In software development, this is usually called the "demo" and is performed by executing the software with constructed test cases that are known in advance to work.

Rigorous testing, on the other hand, is based on observing the difference between the actual behavior and the expected behavior of the software to be tested (the hypothesis). Testing should be seen as both verification (testing against specifications) and validation (testing against the real world). Both verification and validation are needed because specifications aren't perfect.

Aspects of the Science of Software Testing

Pre-definition of Expected Results

Pre-definition of expected results is similar to the scientist that predicts the outcome of an experiment before it is performed by proposing a hypothesis. There is something about predicting the outcome in advance that adds a degree of rigor to the findings. If you wait until the experiment is over and try to interpret the results in light of your understanding and observation, it is easy to convince yourself and others that what you observed was a validation of your hypothesis. When the actual results of the experiment do not match your pre-defined expected results, the discrepancy should lead you to question the experiment, the hypothesis, or both.

Observation

Without observation, it is impossible to tell the outcome of a test or an experiment. Although this makes sense, it is tempting to design tests and experiments that are difficult if not impossible to observe. We may want to prove or test something, but real-world constraints prevent constructing an accurate experiment. That's why you can't test everything – not everything is testable.

Repeatability

In science, an experiment may be performed thousands of times before a trend can be established. The first time a result is observed the scientist isn't sure if the result was due to an unknown aspect of the experiment or a predictable behavior of the subject. To provide a confirmation of the experiment, it may need to be repeated many times. Likewise, in testing, when a defect is observed, the first test may be seen as the indicator and follow-up tests may be seen as the confirmation. After a defect has been fixed, the test must be repeated exactly as before to ensure the fix works.

(Continued on Page 5)

Links

In recent issues of the Software Quality Advisor newsletter (February and March, 2002), we explored how an organization could tell how they were doing in other ways than using the CMM framework. However, I still like the CMM are in response to some requests , here are some links that you may find very helpful for assessing yourself and for learning more about the CMM and related frameworks.

A comparison between the CMM and ISO 9000

http://www.cs.njit.edu/~axp9532/cmm/cmm_iso_compare.html

Statistical Process Control in Level 4 and Level 5 Organizations Worldwide by Ron Radice

http://davidfrico.com/radice00b.htm

Software Acquisition Process Questionnaire

http://www.sei.cmu.edu/pub/documents/97.reports/pdf/97sr013.pdf

For a view of where the CMM started

http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf

Evaluation checklists for CMM levels 2,3,4, and 5

http://davidfrico.com/seievaluationxls.htm

http://davidfrico.com/sw -cmm-checklist.pdf

CMM Questionnaire

http://www.sei.cmu.edu/publications/documents/94.reports/94.sr.007.html

Quotes

"The greatest use of life is to spend it for something that will outlast it."
William James (1842 - 1910)

"Imagination is more important than knowledge..."
Albert Einstein (1879 - 1955)

"Imagination is the beginning of creation. You imagine what you desire, you will what you imagine and at last you create what you will."
George Bernard Shaw (1856 - 1950)

"You cannot depend on your eyes when your imagination is out of focus."
Mark Twain (1835 - 1910)

"Diplomacy is the art of saying 'Nice doggie' until you can find a rock."
Will Rogers (1879—1935)

Great services are not canceled by one act or by one single error.
Benjamin Disraeli (1804 - 1881)

"Formal education will make you a living; self-education will make you a fortune."

Jim Rohn

"A man who lacks judgment derides his neighbor,
but a man of understanding holds his tongue."
Proverbs 11:12, The Bible

Questions From the e-Mail Bag

Q: "What is Defect Density? What is it used for and the basic definition?"

A: Defect Density is a metric that indicates how many defects are in software. This is usually computed by counting the number of defects found in testing the software and dividing by a sizing measure such as function points, lines of code, or testable requirements.

Example: 100 defects in a software application with 200 function points would yield a defect density of .5.

Q: I need some clarification on the following words, if you can please define them for me. The first 3 are used interchangeably within my company. I know that they are 3 different things but no one seems to know what exactly each one is.

1) PERFORMANCE TESTING
2) STRESS TESTING
3) VOLUME TESTING

A: Here's my definitions, which I also benchmark against the way most other people use the terms:

Performance Testing - A term that generally speaks to testing an application to validate performance levels or system efficiency. This term also can include load

(Continued on Page 5)

Questions from the Mail Bag (Continued from Page 4)

testing, stress testing, and the testing of throughput levels (volume testing).

Stress Testing - Testing a system to its failure point by applying concurrent user load or very high numbers of transactions. Another aspect of stress testing (although it is not commonly referred to as such) is testing the input a data element can accept or store - essentially stressing the field level elements.

Volume Testing - The volume of data (throughput) that an application can process in a given period of time. (e.g., transactions per minute, per hour, per day, etc.)

Load Testing - You didn't ask, but it fits in - Testing with specified levels of users or transactions. This is also sometimes called Concurrency Testing due to the use of actual or virtual concurrent system users.

Q: My team is responsible for testing mainframe, GUI and soon web based applications. We are discussing changing the name of our team to more closely reflect what we do. We're considering Quality Control and Quality Assurance. What is the difference between control and assurance?

A: QC is testing or inspecting something for the purpose of finding defects. QA is the management of quality and oversees the adherence to processes. QA also includes measurement, standards and process deployment besides being consultants to the test team. Realistically, most QA groups perform some level of testing, but in the classic definition QA is a facilitator to testing.

"Realistically, most QA groups perform some level of testing, but in the classic definition QA is a facilitator to testing."

The Science of Software Testing (Continued from Page 3)

Although this sounds simple, it may be very difficult in actual practice to get the second test environment set up exactly as the first test environment.

Construction of the Experiment

In scientific research, experiments are carefully planned and controlled. The laboratory environment grew out of the need to prove conditions during an experiment and to repeat the experiment. In this analogy of testing as science, what many people do is perform experiments in their kitchen, not in a controlled laboratory environment. This is a critical lapse, as the test environment can impact many external and internal factors of the test which could very well lead to false test results. I would venture to say that no other discipline could get away with the lax methods used in many software tests, especially where environments are concerned.

In testing, carefully constructed and controlled environments are sometimes needed to get the level of test reliability that is appropriate to the risk. Your test is only as good as the test environment!

Performing the Experiment

The performance of the experiment is an exercise in carefully following the design of the experiment. The research scientist doesn't improvise unless they are doing work apart from the plan. Granted, some of the great scientific discoveries have occurred because the researcher tried something other than the planned experiment, but these are exceptions rather than the rule. In testing it is important to stick to your test plan. It's alright to test other cases, just be sure you document what you did so you can repeat the test if you have to.

Having a Control Group

In scientific experiments control groups are used as a baseline for comparison of results. For example, a researcher might test a trial medication on one group of people while giving a sugar pill (placebo) to another group. The people in the experiment do not know if they have been given the real medication or the placebo. This "double blind" research helps to counteract subconscious biases.

In testing we also need a baseline of correct system behavior as a baseline.

Interpreting Results and Drawing Conclusions

One of the great challenges of science is to observe the tests of a hypothesis and make an objective reasoned interpretation of the results. The challenge of doing this task is maintaining objectivity and having the courage to report what you actually observed as opposed what someone else expected to see. Gee, that sounds familiar.

In testing, you can only speak to what you have observed. It is unrealistic and unwise to predict results from what could be seen from tests not performed.

Modifying the Hypothesis

In testing, the main hypothesis is often that the system should work under given conditions. However, there is another opposing hypothesis that although the system should work, there are defects in it that need to be found. The second hypothesis is the safest one.

When your test results prove the second hypothesis is true, then a fundamental shift starts to occur in the minds and attitudes of those who held the first hypothesis. This is when many people instead of modifying the hypothesis try to discredit or invalidate the experiment or the person performing the experiment.
(Continued on Page 6)

Keys to Successful User Acceptance Testing (Continued from Page 3)

keys to guide users in planning their test.

Key #1 – Understand that UAT is Performed at the Worst Possible Time in the Project

Most UAT efforts happen at the end of the project because this is when the entire system is assembled or installed. Until the end of the project, users may be able to test parts of the system or application, but not the system as a whole. This is bad because the end of the project is the worst time to find and fix major problems. Each problem found and fixed in system testing or UAT which has been in the system since requirements has a 10 times or greater cost factor had it been found or fixed in requirements or design. This is due to the ripple effect that the fix may cause in other areas of the system.

The solution is to involve users throughout the project from the very beginning. When users are providing input to user requirements, they can also be defining acceptance criteria and can be involved in requirement reviews and inspections.

Key #2 – Base the Test on Real-world Conditions, Not User Requirements

This is one of the most controversial things I teach about UAT, but if you don't base the test on real-world conditions you are missing the point of UAT. There are two sides to testing – verification and validation. Verification is testing based on specifications and requirements and is often performed by the producer of the software. Validation is testing based on real-world operational conditions and is often performed by the user or customer. Both perspectives of testing should be performed. Validation is necessary because requirements have defects. In many cases, the requirements are not available, such as in the case of vendor-developed software.

The solution is to have users design tests that model their world. The test is to determine if the system or application will correctly support the real world conditions.

Key #3 – Understand Your Users

UAT is not a "one-size fits all" activity. Some user groups will not have the motivation, time or skills to design and perform an adequate test. Some user groups will be able to perform a very extensive test. One solution is to profile the affected user groups. I often categorize users as:

- **Not motivated or skilled** – These people may not be against the UAT effort, but they just may not be aware of how to perform UAT or understand the importance of UAT. These users may lack basic computer literacy or management support to allow them to participate in the test. This is most often where surrogate users may be needed. In addition, the tests may be minimal at this level.
- **Somewhat motivated, but lacking skills** – At this level you have a chance to get user involvement but will have to build testing skills. Tests may range from low to moderately complex.
- **Somewhat motivated and skilled** – At this level the skills are in place, but you will have to identify motivating factors and market those to the prospective testers.

(Continued on Page 7)

The Science of Software Testing

(Continued from Page 5)

This often takes the form of blaming the testers for the defects, which is like blaming a research scientist for the results of a correctly performed experiment.

However, let's say for a moment that people reach agreement that the first hypothesis was wrong and that the software does have defects and needs to be fixed. This may imply that the software will be delivered late and other people will be held accountable. Although these consequences may occur, people need to face reality and correct the problems instead of focusing on their own agendas.

Perhaps this aspect of testing is most closely aligned to the science we see practiced today. If the research confirms the hypothesis, we hear about it. If the research supports a contrary hypothesis, especially one that goes against conventional beliefs, those findings may never be published.

The Longer View

The reason scientific research is performed is to explain the way observable nature behaves. I would also add that a great benefit of that knowledge is to improve things we currently do. It has been said that the thing that distinguished Thomas Edison from other inventors was that he always had a keen sense of how science could help people by improving their lives. Edison also had a good sense of business as well. He knew when to stop research and build the project.

In testing, the initial goal is to find defects. However, that is a short-sighted view and fails to make the best of the resources that have been expended on creating and fixing the defect. The longer view of testing is to build ways to prevent similar problems in the future by improving the processes used to build the product.

I believe we are far from seeing software testing performed as a scientific process, but it gives us something to think about and relate to, especially when it comes to evaluating the rigor and reliability of test results.

Conclusion

There are many points in common between software testing and traditional science. In fact, software testing may be closer to a science than to anything else we can relate. These similarities can be helpful in understanding testing and explaining testing to others. The similarities also provide a benchmark of how rigorous a process we are using in defining and performing testing processes. Although not every test will need to be performed at the rigor of scientific research, some tests need to be performed at that level because of high risk.

"In testing, the initial goal is to find defects. However, that is a short-sighted view and fails to make the best of the resources that have been expended on creating and fixing the defect."

Keys to Successful User Acceptance Testing

(Continued from Page 6)

Tests may range from low to high complexity.

- **Very motivated, but lacking skills** – The users in this group are ripe to learn new things and contribute to the project. That's a good combination! Good training will often complete this picture. At this level, these people may perform tests in the minimal to moderately complex range.

- **Very motivated and skilled** – These people are like "instant testers." Your training efforts will likely be minimal with this group and you shouldn't have to spend a lot of time motivating them. The challenge is to use their talents wisely and not burn them out during the test. These people can perform a full range of tests from low to high complexity.

Key #4 – Involve Your Users

Some organizations perform UAT with surrogate users – people who take the role of users but who are not the actual people in the field that will eventually use the application. The risk with this is that when the system is deployed, the real users will find problems the surrogates didn't consider. I only recommend this approach when the actual users are unavailable or unwilling to participate in the test. Even then, the risks are high.

The solution I advise is to at least hold limited review sessions with the actual users. Complete review sessions which examine items such as user requirements in detail are even better. Also, have contingency plans in place when unexpected problems are found during UAT. If users are unwilling or unable to participate in the project, raise this situation as a risk in the project status reports.

Key #5 – Match the Intensity of the Test to the Relative Risk and the Skills of the Users

Not every project requires extensive testing. However, for those projects that control high levels of assets or affect personal safely, extensive validation is required. Users and others on the project may question the need for defined test cases and test scripts, but when viewed in the light of project and business (or operational) risks, the time and resources spent in effective testing are resources well spent.

To match testing to the risk, perform a risk assessment that can be quantified and documented. Just guessing at the level of risk is not good enough to explain after a critical failure why you thought something was a low risk of failure. The risk assessment should indicate which system and business areas are the most exposed to risk. This allows test resources to be allocated where they will have the greatest impact to detect defects that may have severe negative consequences.

Key #6 – Plan the Test in Advance

There are three levels of test planning (Figure 1):

- **The strategic level**, which specifies what is to be performed in testing. The test strategy describes high-level direction and objectives, but stops short of "how" the test will be performed.

- **The tactical or logistical level**, which is also considered high-level, but describes how the test will be performed. This is basically the project plan for the test and is often written to focus on one phase of testing, such as unit, integration, system, or UAT.

- **The detailed test case or test script level**, which defines in detail the actions to be performed, the expected results and the procedures to perform the tests.

Some people prefer to use informal and random approaches to testing, which can find obvious defects but will often miss the more deeply embedded defects that are often the most troublesome. Another problem with a lack of test planning is that you never quite know for sure when you have completed the test. A test plan contains measurable objectives and pass/fail criteria. Detailed test plans also make it possible for one person to design the test and for many people to perform it. Finally, detailed test plans give you a way to recreate a test if you have to perform it again.

Once again, the extent of your test planning effort should be reasonable in light of the relative risks.

Key #7 – Review Your Test Plans

Test plans can have errors just like any other type of project documentation. UAT plans can be reviewed by the UAT team, a Quality Assurance (QA) team or facilitator, the project manager, or any other people with know ledge of testing and the project.

Conclusion

User acceptance testing is not trivial or easy. UAT can be one of the most critical and risky types of test on a project, which means that a great deal of care should be taken when planning, executing and evaluating the results of UAT. These keys of UAT have worked for other organizations in planning and performing UAT and they can work for yours as well. There are also more keys of UAT that are out there that perhaps I can explore in future articles. In the meantime, if you have any great keys to UAT, let me know at rrice@riceconsulting.com and I'll be glad to include your contribution in a future newsletter.

"Some people prefer to use informal and random approaches to testing, which can find obvious defects but will often miss the more deeply embedded defects that are often the most troublesome."

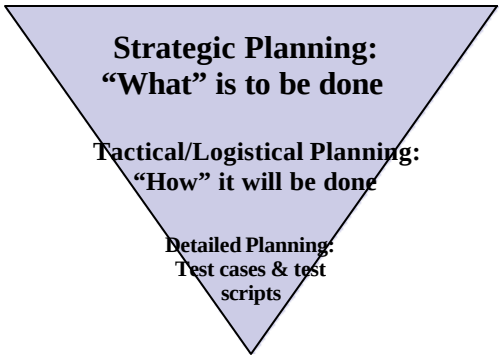

Figure 1—Levels of test planning detail

"I write down everything I want to remember. That way, instead of spending a lot of time trying to remember what it is I wrote down, I spend the time looking for the paper I wrote it down on."

Beryl Pfizer

The Software Quality Advisor Online

May, 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

June 2002 Issue:

Coming to Chicago!

August 14—16, 2002

PMG RCS Consulting Excellence!

User-oriented Practices for Delivering Quality Software

Presented by Process Management Group, Ltd. (www.pmgltd.com), the Midwest's Premier Provider of IT Quality and Software Testing Services, and Rice Consulting Services, Inc. (www.riceconsulting.com), a world recognized leader in Quality and Testing Training.

See details and register at
www.riceconsulting.com/chicagoq3_2002.htm

Calendar of Events

KCQAA Spring Conference

Kansas City, Mo, June 10—12, 2002

Randy will be presenting a three-day program on *User-Oriented Methods for Software Quality*

www.kcqaa.org

User-oriented Practices for Delivering Quality Software, Chicago, IL, August 14—16

Sponsored by the Process Management Group, Ltd. And Rice Consulting Services, Inc.

www.riceconsulting.com

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.