

Software

Q U A L I T Y

VISION:

To be the leading authority and recognized champion on issues related to software quality.

MISSION:

The Software Division is the community of individuals and organizations committed to: Seeking and applying technologies, concepts, and techniques to improve the quality of software products, processes, and services; growing their professional skills; and building a stronger level of professionalism in that community.

Check out the
Software Division Web site at
www.asq.org/softwareforum/

The contents of this publication are protected by the copyright of the American Society for Quality.

Interested in reproducing this information? See ASQ's guidelines for using copyrighted material.

<http://www.asq.org/join/about/copyright/index.html>

ESSAY: PROGRAMMING TECHNIQUES APPLIED TO TEST AUTOMATION

BY ALEX HILGENDORF

Why create test automation?

Many project teams have uncorked celebratory champagne bottles while their software was being rolled into production, only to wonder if their software would actually work. The decision to release a product is, if not the most important decision, then at least the most visible decision that a project manager makes. Production costs due to poor software are large. The cost of delaying a project is large as well. Somehow a balance must be found allowing customers to receive appropriate solutions in a timely manner. And making the right release decision requires having enough of the right information.

Assume that code in a shared library has been changed late in the development process. This could have a large regressive impact on much functionality. The project manager may only know that 40 person hours have been dedicated to searching for regressive defects and that two important defects were found. Without any past results to compare against, "40 hours of testing" has little meaning. Perhaps another 40 hours of testing would find additional regressive defects. Also, without a list of test cases, there can be no understanding of how well the testing effort spanned the product. Perhaps the testing effort focused on test cases that span half of the specifications impacted by the change to the shared library. This project manager truly has too little of the right information.

Software testing is just one of many activities that can provide information to support important decisions. But often the testing effort does not supply information fast enough and does not supply information that spans the product. In fact these two issues can even be related. An eager project manager may decide to release a product before all test cases have been investigated under the assumption that additional testing will cost more than the benefit of the additional investigation.

In the testing industry we have advanced our ability to create and manage requirements driven test cases that yield a broad picture of the quality of our products. And in an effort to reduce the time that we spend executing our test plans we have spent large amounts of money on off-the-shelf test automation products only to find ourselves performing all of our regression testing by hand just a year later. In this case we might better save our money for the release parties at the end of our projects.

Automation prerequisites

The human and technology environment required to leverage automated testing toward project success is greatly misunderstood. In fact not every project can benefit from test automation. But if there is to be any benefit at all, then three factors need to be present:

(cont. on p. 3)

OUTGOING CHAIR'S CORNER

BY MICHAEL KRESS

December 2004

As I surrender the Software Division chair to our new chair, Doug Hamilton, I am mindful of the ever-changing landscape in which software quality practitioners fit in today's world. Whereas years ago, most systems were designed from "scratch" within the framework of the "requirements-design-code-integrate" chronological process model, systems design in today's world is a bit different. New methodologies and strategies such as EXtreme Programming, Agile methods, Model-based-development, and component engineering are getting more and more attention.

The building of systems using large amounts of "COTS" software has given rise to a new awareness of the importance of COTS software to business, and in some cases, to safety-critical applications. Just last week a hospital made headlines as being in financial trouble because of faulty billing software, which caused them to lose "tens of millions of dollars" in delayed payments.

Work is currently under way within the ISO/IEC JTC1 SC7 community to address a wide range of standards for COTS issues from packaging information, user documentation, product descriptions, and the usability testing. These standards (such as ISO 12219 and 9127, for example) are aimed at what I like to think of as a *Consumer Reports* for software. The full disclosure of the features of the software, capabilities, limitations, and applications, with emphasis on metrics for the efficacy of the package, are being addressed.

I just returned from a workshop of SC7 held at the University of Granada in Granada, Spain, where these issues were discussed. ISO 12119 is in revision addressing issues of user documentation that include, but are not limited to: completeness, correctness, consistency, understandability, ease of overview, functionality, reliability, usability, efficiency, maintainability, portability, and quality-in-use. Test documentation is also heavily addressed with requirements for the test specification, test environment, test case descriptions, test procedures, test results, execution, and anomaly reports. Much of what is included is being harmonized with the software measurement standards of the new SQuARE Model series 25000, (Software Quality Product Requirements and Evaluation).

I wish to express my gratitude to the Software Division Council and its support infrastructure at ASQ headquarters for their support over that last two years. These were tough years for us with declining attendance at our conferences due to 9/11, cutbacks, international travel restrictions, and a proliferation of conference choices. Our next conference is scheduled for March 21-23, 2005, in Orlando and promises an exciting agenda including an aerospace panel composed of well-known software quality practitioners from industry, academia, NASA, DOD, and the FAA. Please visit our Web site for details.

On a closing note, I wish to extend my support to our new incoming chair, Doug Hamilton, who has worked tirelessly for many years as our certification chair. Doug is one of the pioneering members creating and maintaining the highly popular Certified Software Quality Engineer program. Thank you to Doug, the rest of the council, and all of you members of the Software Division. We hope to bring you news and resources helpful to your job and personal objectives in the years ahead.

CHAIR'S CORNER

BY DOUG HAMILTON

Thoughts and Volunteer Opportunities

My name is Doug Hamilton and I am the Software Division chair.

The 14th International Conference on Software Quality will be held March 21-24, 2005, in Orlando. I hope to see many of you there. For more information, visit <http://www.asq.org/softwareforum/conferences/>. The conference is a great opportunity to learn, talk to other software quality professionals, and establish professional contacts.

Software is a fascinating industry to work in since everything is changing so rapidly. The way I stay up-to-date is through my volunteer activities with ASQ. I always learn something at CSQE workshops and any ASQ meeting I attend by hearing the experiences of other software quality professionals. Our volunteers come from many industries, geographies, and backgrounds so if you are interested in helping, let me know. We can always use some new ideas and fresh energy !!! Also, please pass along any ideas you have for improving the division and making it more valuable to our members.

The entire Software Division Council joins me in wishing each of you, your families and loved ones a safe and prosperous new year.

PROGRAMMING TECHNIQUES

CONTINUED

Management support: There is a large overhead in setting up an automated test environment. As with any process improvement, management has to be committed to providing time and training.

Domain knowledge: Let it be known that any person who will create effective test automation must have enough domain (industry specific) expertise to understand the test cases and to interpret results.

Technical skills: Many companies succeed in their manual testing efforts because they are able to teach testing theory and techniques to employees who are already domain experts. But without additional technical training, these newly created manual testers are not able to create test automation that stands the test of time.

The first two issues listed above are worthy of their own essays. But the remainder of this paper will focus on the application of technical skills to test automation development. First we will define what traits are found in a good automated test. Then we will show that the same techniques that create good computer programs also create good test automation.

Quality traits of good automation

On any given day the value of an automated test lies in the information that it will give me if I choose to execute the test. If the module under test operates correctly, then the automation should indicate this. If the module does not operate correctly then the automation should indicate this as well. *But it should also be intelligent enough to know if the test is inconclusive.* The later point makes automated testing unique from manual testing and is often overlooked. A manual tester can usually identify a typo in a manual test script while he/she is executing the script. Automated test scripts that contain typo will often create false positive results that go undetected. This is dangerous.

An automated test should execute in a reliable fashion. If it needs to navigate through four screens/windows before getting to a screen/window that is directly related to its test case, then it should not be overly sensitive to small changes in the layout of the first four screens/windows. For example if the text in the title bar changes and if this is not within the scope of the test then the script needs to ignore the title bar while navigating. Without this intelligence, test automation needs to be updated (fixed) often and no longer offers information in a timely manner.

A test script needs to be tractable. There should be documentation elsewhere for the test case, but the script it self also needs to be readable. From time to time a script will need small changes to allow it to adapt to a changing user interface. If a tester cannot quickly understand what a script is trying to accomplish, then the changes to the script become time consuming.

A test script needs to be extensible. Testers know that no test plan will cover every test case. So a testing environment must be continually expanding to cover additional use cases. Depending on how well a test script is written, it may be used as a basis for additional test cases. But this requires some planning on the part of those who create the original automated test.

A test script needs to persist through the product life span. Some products span several project releases with only minor

changes to the user interface. Due to the high cost and extreme user interface sensitivity of automated testing, these products have the most to gain from automation. In some cases portions of automated test scripts can be used for over five years. Generally, the longer a script can be used, the greater the return on its development costs.

Five traits that an excellent automated test should contain

Informative	Provides information regarding the quality of a portion of the product that is under test
Reliable	Executes reliably without a significant amount of aid from an automation operator
Tractable	Expresses its purpose and methods to a tester in a natural manner
Extensible	Adaptable to an increasing scope of test cases
Persistent	Continues to provide product quality information through and beyond the project life cycle

Programming techniques that create these traits

So good test automation is Informative, Reliable, Tractable, Extensive, and Persistent. These also are traits of good computer programs.... End users want reliability in their solutions. Developers who are fixing and expanding programs like tractable and extensible code. Managers like persistent code because they do not have to pay for redevelopment. And it can even be said that good programs are informative...when they fail, they should be able to communicate their state during the failure.

Now consider the effect of the following programming best practices when they are applied to the development of test automation.

Don't repeat code: Consider a suite of 100 scripts that enter the same username and password but then perform 100 different test cases. What if enterprise security restrictions require that passwords be changed monthly? The solution is to create a common login routine that all scripts use. This way only one routine has to be changed. In fact when common routines exist in a shared library, then the amount of time required to write a new script is dramatically reduced.

Make code easy to reuse: To assure that the appropriate routine can be quickly identified and included in a new script, similar routines should be grouped together in common libraries. When a script is developed to test a specific module, the libraries for that particular module can be included. When multiple automation developers are working on the same product, they often don't know what scripting problems have already been solved. Nicely documented routine libraries will prevent them from redeveloping existing scripts.

Add error checking: Obviously a script must eventually check for an expected output. This type of checking identifies system defects. But a script should also check the preconditions for the test case. If the preconditions are not met, then the test is inconclusive and a testing analyst needs to be informed. Also, if a script fails to create or check for the

(cont. on p. 4)

test case preconditions, then the script likely needs to be fixed or updated. An intelligent script is able to determine how many steps were successfully completed before the failure occurred and is able to communicate this to the automation developer.

Use templates: One or more common templates can be created that are copied to create new scripts. The template can use a format that is easily readable by all testers. This will help automation developers fix and update scripts that they did not design themselves. Also a template can contain code that is used by nearly all scripts that are specific to a certain product. For example a call to a login routine that is used by all scripts can be included in the template. This reduces the amount of mindless time that a developer spends on scripting details. Finally, a good template serves as a learning tool for junior engineers or engineers who are new to the automation tool.

Don't rely too heavily on Wizard code: A script generator records the key strokes and mouse actions of a user while the user interacts with the system and then turns the recording into a script. This can be a fast way to create a script and is easy for a novice tester to perform. But there are many problems. Usually error handling and calls to subroutine libraries must be added manually. Script generators do not economize code because they can't detect when a user is repeatedly performing a complex task. Thus loops must be added manually.

Put abstractions in code, details in metadata: The code within a test script should represent a procedure or a set of steps that governs how interact with the system. This procedure will support many different test cases. For example, a procedure can enter a customer name address and other attributes into a record. But this procedure/script can support many test cases (a person who has two middle names, no middle name...). To aid in the reuse of the script, the details or the supporting data should be extracted from the script. Perhaps the details can be defined in contents at the top of the script. Or perhaps the details can be read from a data table. In fact the data for a script can be exported directory from a written test plan, making the test environment more adaptable to changes.

Master your editor: The difficulty of solving the day-to-day problems of an automated testing environment lessens as a developer learns to effectively utilize their editor and there development environment. For example, a developer may need to quickly identify the complete list of scripts that attempt to access a specific record. Or they may need a way to make several complex changes to a set of data tables so that the same change is made to each table. Advanced editing skills can greatly reduce the amount of time spent on development and can increase the accuracy of script changes. The use of a source code control application might be considered at a method of managing test scripts along with other test documentation.

Master a second language: The automated testing tool that you have chosen is likely designed to work well with your application. But it may not work as well with other parts of

your test architecture. For example, the same tool that can effortlessly drop keystrokes into a user interface may not be the best tool for moving files around in a file system. Consider mastering a second language that will solve problems that your primary testing language does not easily handle.

Verify that it works, validate that the correct test case was implemented: By the time development of a test script has been completed an eager developer may anxiously move on to the next development task without testing the automated test. Before a script is truly finished, its ability to report failure in the presence of a failure and success in the presence of a healthy system must be demonstrated as best it can. This may require adjusting the testing environment so that the script is tricked into thinking that it failed. Finally, the initial test case must be reconsidered so that the developer is confident that the correct test case is being tested.

Document the test automation: Updates will be made to test automation for a number of reasons. System updates will require script updates. New testers will maintain old scripts. If script updates are not made carefully and with appropriate insight, the script will begin to test something other than the intended test case. To mitigate this risk, test scripts should have documentation within the script that describes the test case and the steps the script is performing. When necessary, the script's notes should cross reference external documentation.

Final thoughts

With the proper support and skill set, your automation project will hopefully succeed. Be aware, however, that automation is not right for every project. The nature of the product itself will impact your ability to automate testing for it. Changes to the user interface will cause scripts that navigate through the user interface to report false failures. Also, the cost of initial development can be quite high which is why continual management support is so important.

In conclusion, success of an automation project is dependent on the ability to understand the system, to win top-level support, and to quickly create automation that will provide lasting results. The understanding of the programming concepts needed to quickly create the automation may be what your own automation project is lacking. If so there are many educational resources that can be used.

Tech schools and local colleges offer programming courses. In fact they often do a better job of teaching the same theoretical aspects that vendor supplied courses gloss over. Also, there are an increasing number of online courses that are available. Finally the cost of a good text is small compared to the benefit of the knowledge it imparts and is yet more affordable than a college course. The Andrew Hunt and David Thomas book *The Pragmatic Programmer: From Journeyman to Master* is intended for programmers and yet is wonderfully applicable to test automation development.

Alex Hilgendorf is the certification chair for ASQ's Certified Software Quality Engineer exam. He has worked in the field of software quality since 1997. Currently he works as a developer of test automation at Unity Health Insurance located in Sauk City, WI. He may be contacted at 3319 Harvey Street E, Madison, WI, 53705, 608-273-4679, abilgendorf@voyager.net.

CALL FOR ARTICLES

The *Software Quality Professional* journal, sponsored by the Software Division, is looking for articles relevant to the CSQE Body of Knowledge. The CSQE Body of Knowledge (BOK) includes General Knowledge, Conduct, & Ethics, Software Quality Management, Software Engineering Processes, Program and Project Management, Software Metrics, Measurement & Analytical Methods, Software Verification & Validation, and Software Configuration Management. See <http://www.asq.org/cert/types/csqe/bok.html> for more details. The BOK is very comprehensive so many topics are applicable to the journal audience. Take a look at the Web page <http://www.asq.org/pub/sqp/> to see what the journal is about. The author guidelines are online also: <http://www.asq.org/pub/sqp/author/>. If you would like to discuss a possible journal topic, please e-mail the editor at Sue_Carroll@bellsouth.net.

3RD WORLD CONGRESS FOR SOFTWARE QUALITY

September 26–30, 2005, in Munich, Germany

The 3rd World Congress for Software Quality in 2005 will bring together the leading experts from academia and industry to share ideas, insights, experiences, and advances in software quality, software process improvement, and software development methods.

The World Congress for Software Quality is scheduled every five years and hosted alternatively by the main organizers: the ASQ (American Society for Quality) Software Division, JUSE (Union of Japanese Scientists and Engineers), and the EOQ (European Organization for Quality). The first World Congress for Software Quality was held in San Francisco in 1995, the second in Yokohama in 2000, and the third will be held in Munich.

The 3rd World Congress for Software Quality runs in the same time as the original “Oktoberfest” at Munich in 2005. The World Congress is scheduled for five days (September 26–30). These five days will offer tutorials, workshops, panel discussions, paper and keynote sessions, and an exhibition of software tool suppliers as well as excursions to the research laboratories of the main sponsors. Social events, receptions, and other opportunities to meet experts and VIPs in the software engineering business will be provided.

The Congress Web site is <http://www.3WCSQ.org>.

Patricia McQuaid is the chair of the Program Committee for the Americas and can be reached at pmcquaid@calpoly.edu. If you send an inquiry, please place “3WCSQ” in the message header.

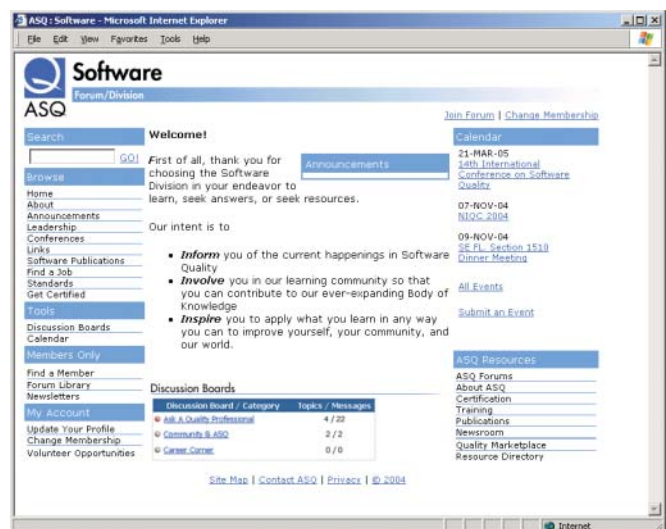
ASQ SOFTWARE DIVISION MEMBERSHIP UPDATE

**HANK SOBAH, VICE CHAIR,
MEMBER SERVICES,
ASQ SOFTWARE DIVISION**

As of December 31, 2004, the Software Division had a total of 3,837 members. A new membership category, the Forum/Division category, has 30 members, 21 of whom are new ASQ members this year. Roughly two-thirds of the 30 Forum members are new to the Software Division, with the remainder appearing to have changed membership categories over the past year. The division has 25 members who are ASQ Fellows; 302 Senior members, 75 of whom achieved Senior level over the past year; and 345 Sustaining members.

Throughout the year, the division has been working to replace and improve our Web site. Many thanks to the people who have worked hard to make this happen, and especially to Pablo Baez, ASQ Community Care administrator, ASQ headquarters, who did a yeoman’s job in putting the new site together for us. Please see the sample home page below, take some time to visit our new site at <http://www.asq.org/softwareforum> and let us know how you like it. We’re always open to suggestions and/or new ideas.

As always, thanks to everyone who is supporting the advancement of software quality processes and quality awareness.



SYNOPSIS OF THE TWENTY-SECOND ANNUAL PACIFIC NORTHWEST SOFTWARE QUALITY CONFERENCE

MIKE KRESS
ASQ SOFTWARE DIVISION
REGION 6 COUNCILOR

Oregon Convention Center
Oct. 12–13, 2004
Portland, OR

Abstract

As an attendee and participant at this conference for the past 11 years, I attend and report on selected presentations and attempt to provide a synopsis the papers for our readership. The report continues a tradition begun by a retired colleague, Neil McQuage, who, in the 1980s read every paper in the proceedings and reported on the most relevant ones to the Boeing Company. Although I do not read every paper and cannot attend every track, I have herein selected certain papers for this report. The proceedings from this conference cover 44 presentations in 569 pages. The presentations I elected to attend focused on the growing implementations of extreme programming and agile methods, metrics, TQM, initiatives on lessons learned, and requirements management.

I vigorously endorse this annual event as great value for your training or continuing education dollar. It attracts speakers and attendees from all domains within the software arena including developers and representatives of international regulated and nonregulated software commodities

Agility for Testers

Elisabeth Hendrickson

Elisabeth Hendrickson is an independent consultant specializing in software quality and testing. She's become involved with the eXtreme Programming community in the last year and is convinced that Agile processes have a great deal to offer the software industry. With more than 15 years' experience working with leading software companies, Hendrickson has seen software—and thus software quality—from numerous perspectives. She has been a tester, test automator, technical writer, programmer, help desk technician, system administrator, and manager (sometimes simultaneously). An award-winning author, she has numerous published articles and is a frequent speaker at major software quality and software management conferences.

Agile Methods has become an umbrella term for a collection of methodologies that increase agility, including Extreme Programming (XP), Scrum, Crystal, and Lean Development. Agile teams accept change as inevitable and tailor their processes accordingly. Short iterations, frequent feedback, continuous integration, “merciless refactoring” (fixing and improving the code) prevent the code from becoming fragile over time.

In traditional testing, testers typically push for an early code freeze so they have time to test the feature-complete system. Agile testing takes continuous change into account.

Some the differences from traditional development are:

Traditional	Agile
Strict change management	Change is inevitable
Up-front planning	Plan to next iteration
Formal entrance/exit criteria	Collaborative
System/regression test	Detect defects early
Documentation intensive	Working software

An informal poll of 135 testers across 57 organizations revealed that one-third of their time was consumed in documentation. In Agile, tools like wiki* are used to streamline documentation.

In Agile, the tester role is changed from that of the customer's “last line of defense” (I'll protect you) to one of supporting and contributing to solutions.

In Agile, you don't get “error-free” code, you get “good enough” code. “Good enough” typically means free of major bugs, crashes, corruptions, things that would prevent usage. The rationale is that many customers are satisfied with “good enough” software.

*[*Editor comment: A “wiki” (named for the Hawaiian word for ‘quick’) is a collaborative Web site users can create in a snap and invite anyone else to edit or add material. Ref. BusinessWeek Oct. 18, 2004]*

Demystifying Microsoft's Quality Assurance Process

Ambrosio Blanco

Microsoft Corporation

Ambrosio Blanco joined Microsoft in 1992 as an API tester on Microsoft COBOL. Since then he has worked on Microsoft FORTRAN, integrated mathematical and statistical libraries, Office 97 and 2000, and Internet Explorer 4.0. He is currently a test architect in the Knowledge and Interchange incubation team in the Information Worker division. He received his degree in computer science from the University of the Philippines in 1990.

Blanco began by conceding that early perceptions of the quality of Microsoft products were largely unflattering. He states, “...negative quality issues garner significant media attention and such visibility has contributed to a general attitude of equating Microsoft products with poor quality and therefore poor quality assurance.” He hopes with this paper to “demystify” this perception.

Microsoft has three distinct organizations in each product team—program management, development, and QA. QA is a “peer” organization with a strong voice in project decisions, scheduling, and feature planning.

The product life cycle consists of staged deliveries or Milestones, M0, M1, M2, etc., similar to the Sashimi model. (The technique is an adaptation of the waterfall model in which phases can greatly overlap.) During the spec and design phases, QA is fully involved providing estimates, and giving feedback on proposed design approaches, including testability and usability. Once all the features are checked in to the source tree, the milestone is considered “Code Complete.” At this point, a number of

testable units are available for that milestone. After Code Complete, the next goal is Zero-Bug-Bounce, (ZBB) a goal of having zero active bugs in the bug database. Once ZBB is reached, the bar of bug fixes is raised significantly in order to reduce code churn; only “showstopper” issues are allowed at this point.

After ZBB, (and customer beta-testing) the code goes in ESCROW (also called “baking”). This is the team’s final window of opportunity to find major bugs or showstoppers. After ESCROW, a formal sign-off process occurs giving all team members the right and responsibility for voting for release. All release criteria are crafted to be measurable and objective; however the human element is considered.

Triage (a term borrowed from the medical field) is a three-person team (tester, developer, and project manager) assigned to determine bug priorities. Triage decides to fix or postpone the fix. Each decision is documented to capture the bug. If Triage cannot reach consensus on an issue, it is elevated to a War Team of senior technical leads and managers.

Near the end of the ESCROW process, the team uses a set of metrics to help decide the worthiness of release. Metrics include:

- Bug count
- Bug slope (a diminishing bug rate)
- Bug types
- Test case count
- Code coverage
- Customer feedback
- Personal impressions of the team.

Microsoft employs a surprisingly high number of methodologies in the build process including:

- Daily builds
- Test passes (on daily builds)
- Unit tests
- Code coverage analysis
- Static code analysis
- Model-Based Testing* (not widely practiced)
- Pair-wise testing (some teams)

The latter two methodologies are not widely practiced yet because of their newness and skepticism surrounding them.

Finally, two noteworthy new initiatives are being instituted at Microsoft:

Engineering Excellence (EE): an attempt to formalize production of software by treating it like a classic engineering discipline.

Trustworthy Computing (TW): Addresses the growing concern for security vulnerabilities of Microsoft products.

*[*Editor comment. The FAA is looking at Model-based Development closely. Ref FAA Tools Forum, May 2004, Orlando.]*

Dynamic Design Analysis: Testing Without Code

Scott Whitmire

Scott Whitmire, a professional software developer for more than 24 years, has used dozens of tools, technologies, and paradigms. He has worked in manufacturing, wholesale

distribution, retail, and accounting domains. He has been a member of Software Engineering Process Group, and has worked, written, and taught extensively in the area of software metrics. He developed 3D Function Points, and wrote Object-Oriented Design Measurement, published by John Wiley & Sons in 1997. Whitmire is currently a lead design and product support analyst for Olympic Distribution Solutions, LLC.

Whitmire opened with an analogy of software to other engineering sciences. The notion of being able to verify or validate a design without having to actually build the design is well known and appreciated in other engineering sciences such as civil and electrical engineering. The ability of a design to withstand stresses and strains under defined loads and usage periods are the responsibility of design engineers using calculations based on the physical laws of kinetics and kinematics, electrical, magnetic, wave theory, etc.

The design and construction of bridges and other major structures offer strong parallels with design and construction of software. Most software developers don’t know if their product will work until after they’ve built and run it for the first time. Even then, they don’t know if they’ve solved the right problem until their customers get their hands on it. Very few of us design our software; even fewer analyze our designs.

In electrical engineering, design analysis tests for radio frequency interference between components, voltage and current potential, physical size, and even the amount of energy released as heat, so these issues can be dealt with in the design. The objective is to find out before a design is built, whether it will fail.

Three important lessons software engineers can learn from other engineering disciplines are:

1. We learn more from failures than from successes.
2. It is easier and cheaper to revise a design before it is built.
3. The earlier in the process we catch an error the cheaper it is to fix.

This paper concentrates on providing a technique to analyze the static properties and dynamic behavior of a software design, or part of one, before time and money are spent writing code.

Software is stressed when it is asked to respond to an event that it did not generate. In this sense, the load that a software system must bear deals with both the types and quantity of events to which it must respond. Normally we need code to test loads. However, we can estimate how our design might perform using algorithmic analysis methods and techniques. A technique based on set and category theory, relational algebra, and formal specification language provides for representing and analyzing the dynamic behavior of the design before time and effort are spent to build it. The technique involves defining an initial state for a scenario and seeding that state with an event that takes the form of a message that originates outside the design. Working through various scenarios, it is fairly easy to determine which messages will succeed and which will not.

Whitmire concedes that the technique does not guarantee that you will find all the errors in your design. You still have to imagine the combinations of external events and starting states that can cause your design to fail. It is likely that you’ll miss one or two.

Like steel and concrete, software code is so expensive to modify that we can no longer afford to build large complex systems several times while trying to deliver it the first time.

(cont. on p. 8)

Retrospectives: The Impact Process Improvement

Debra Schratz
Intel Corp.

Debra Schratz has 7 years' experience in quality engineering. She currently works as a platform quality engineer in the Platform Quality Methods group, part of the Corporate Quality Network at Intel Corporation. Since January 2003, Debra has delivered over 30 project and milestone retrospectives for Intel worldwide. Prior to her work in quality, She spent 8 years managing an IT department responsible for a 500+ node network for ADC Telecommunications. Schratz is a member of the Rose City SPIN Steering Committee. She holds a bachelor of arts degree in management with an emphasis on industrial relations.

A “retrospective” is a method to capture lessons learned from projects. It is the product of a book by Norm Kerth, *Project Retrospectives: A Handbook for Team Reviews*. The methodology encourages the team to spend 2% of the total project to uncover new ways of working together more effectively. The emphasis is on learning, not faultfinding. For a retrospective to be effective, it has to be done in the context that no matter what is uncovered and discussed, it is in the spirit of improvement. Intel had an existing post-mortem practice, but it had the following shortcomings:

- Project managers made poor facilitators.
- No defined process.
- Post-mortems done ad hoc.
- Untimely meetings.
- Issues collected without action plans.
- Non-relevant information collected.
- No consistent format to document findings.
- No central repository for information.

In late 2002, a pilot program was authorized. The approach taken included the following measures:

- Establish a consistent process.
- Hold retrospective within 2-6 weeks of project end.
- Use a neutral, skilled facilitator.
- Create action plans for items agreed to.
- Share best known methods (BKMs).
- Communicate with management.

The program has 3 phases:

Phase 1. Get ready—Surveys were conducted approximately 2 weeks prior. The surveys helped if some members couldn't attend. They also helped bridge language barriers and cultural issues, which might impede sharing openly.

Phase 2. Look at the Past—What worked well? What did we learn? What do we want to do differently next time? What needs more information or detail?

Phase 3. Plan for the Future—Prepare problem statement, recommended solution, identify obstacles, support responsibilities, establish an owner, and set due dates.

A year later, more than 30 retrospectives were completed in various groups at Intel. The main issues coming out of the retrospectives were:

- Poor cross-site/division communication.
- Unrealistic schedules.
- Incomplete requirements.
- Excessive rework.

After a year's worth of retrospectives, the main improvements reported were:

- Improvement issues more visible.
- Reduced rework.
- Better change control.
- Improved product quality.
- Minimized requirements volatility.
- Shortened time to market.
- Easier information retrievability.

The total impact within Intel is still unknown; however, anecdotal reports are very favorable. An intangible result is an enthusiasm among the pilot groups such that they are continuing the retrospectives (voluntarily) after the trial periods. One project manager estimates that one 4-hour retrospective saved his project 4 months of labor. The author's research finds that retrospectives are being used at Siemens, Standard Insurance, and the state of Washington, reportedly with “exciting possibilities.”

The Requirements Dilemma: It's Hard Work

Debra Aubrey
General Dynamics

Debra Aubrey is a practicing systems and software engineer with the good fortune of having worked across the United States and in Europe. As a systems engineer at General Dynamics in Scottsdale, AZ, she divides her time between process consulting and the application of Volere techniques to eliciting, writing, and maintaining requirements in real projects. She brings a wealth of experience and a pragmatic “just do it” approach to requirements management.

Aubrey has worked in a wide variety of industries, including scientific instruments, color printers, consumer electronics, semiconductors, avionics, and radio products. She recently presented a paper, “The Requirements Dilemma: It's Hard Work” at the 7th Philips Software Conference in Eindhoven, the Netherlands. Voted “best paper” by the conference attendees, her paper led to an article, “Defining Requirements—Recipe for Success,” published in Colophon, Philips Semiconductors' consumer segment magazine.

The ability to write good requirements is an often-neglected skill. While tools can help organize requirements, they do nothing about the quality of the requirements. Good requirements must be concise, complete, verifiable, consistent, and unambiguous. Developing requirements can be a daunting task; hence it is oftentimes bypassed in order to “get on” with the design.

This paper describes an approach applied and refined at General Dynamics C4 Systems in Scottsdale, AZ. There are 3 basic steps to the approach:

1. Gather requirements.
2. Document requirements per agreed-to rules.
3. Review and control requirements.

An initial step is getting agreement on a vocabulary.

Requirement: contains the word “shall”

Constraint: contains the word “must” or “must not”

Objective: contains the word “should”

Assumption: contains the word “will”

Requirements may come from many sources:

Competitors

Brainstorming

Legacy requirements

New requirements

Industry requirements

During the requirements gathering phase it is less important to write “good” requirements than it is to capture ideas. Use of a requirements-gathering template allows ideas to be refined later. An important element to requirements gathering is “knowing what you don’t know.” By designing the template with placeholders for additional information, the basic requirement can be captured, recognizing the gaps that need to be filled in later.

Requirements should be uniquely labeled. In specification writing, the normative (requirements) text is sometimes obscured by the informative text (clarification). Normative text describes the “what.” Informative text explains the “why.”

Example:

FRS-R-127 Source: CRS-R-015 (normative)	The source code shall comply with ANSI C standard
(informative)	<i>The product will be sold in either source or binary format. When the source code is shipped, it must comply with ANSI C to help ensure that it will compile using most commercial compilers.</i>

Traceability is another attribute of good requirements management. Traceability provides the ability to directly link each requirement to its origin, and used to ensure that the product is built and tested to meet the customers’ expectations. Traceability is built into the requirements label by explicitly stating the source of each statement. A statement’s legitimacy should be questioned when the source of the statement is missing.

As requirements emerge, it is common to encounter design requirements that cannot be explicitly traced to a source statement. In such cases, a “Rationale Statement” is helpful.

Rationale 002	<i>During product deployment discussions, the need for the product to provide different modes of operation was identified. The product must allow for normal operation, software, and commissioning.</i>
--------------------------------	--

In labeling requirements it is helpful to follow these rules:

Every requirement is assigned a unique identifier.

Requirements are never renumbered.

Requirements numbers deleted are never reused.

It is easy to imagine a sequence of events where a test case is developed to verify a requirement statement that is later deleted. Suppose the deleted identifier is immediately reused. An audit of

the test cases against the Product Requirements could show that the requirement is covered, when, in fact, the test case is not valid for the reused identifier.

Before requirements are entered into the specification, each requirement should be reviewed by an independent test team. A technique known as the “Quality Gateway” conceived by Suzanne and James Robertson, (*Mastering the Requirements Process*, Addison Wesley, Harlow England, 1999) serves as a filter to ensure requirements meet the established criteria and are also “good” requirements.

[Editor comment: It is little wonder this paper was chosen “best paper” at the Philips conference in the Netherlands. It espouses a common sense, methodical, and meticulous way to control one of the most prevalent sources of cost overruns and schedule delays—poor requirements. It explains why it is inadvisable for developers to depend too strongly on tools because of the human element involved in good requirements writing.]

Pairwise Testing: A Best Practice That Isn’t

James Bach
Principal, Satisfice Inc.

Patrick Schroeder
Professor, Dept. of EE/CS
Milwaukee School of Engineering

James Bach is a pioneer in the discipline of exploratory software testing and a founding member of the Context-Driven School of Test Methodology. He is the author (with Kaner and Pettichord) of Lessons Learned in Software Testing: A Context-Driven Approach. Bach started at Apple and went on to work at several market-driven software companies that follow the Silicon Valley tradition of high innovation and agility.

Patrick Schroeder, currently an assistant professor at the University of Wisconsin-Milwaukee, has 14 years’ experience in the software industry, including 7 years at AT&T Bell Labs.

Pairwise testing is a wildly popular approach to combinatorial testing problems. The number of articles and textbooks covering the topic continues to grow, as do the number of commercial and academic courses that teach the technique. Despite the technique’s popularity and its reputation as a best practice, we find the technique to be over-promoted and poorly understood.

Pairwise testing is a form of combinatorial testing. A short cut. Combinatorial testing is difficult because of the large number of possible test cases (a result of the combinatorial explosion of selected test data values for a system’s input variables). Running all possible combinatorial test cases is generally not possible because of the large amount of time and resources required. (Editor: This has been known for many years.)

Pairwise testing is an economical alternative. In pairwise testing, a set of test cases is generated that covers all combinations of the selected test data values for each pair of variables. Pairwise testing normally begins with the selection of values for the system’s input variables. These values are often selected using domain partitioning. (A free, open source tool to produce pairwise test cases is available from Satisfice.)

(cont. on p. 10)

The virtues of pairwise testing are advertised as:

Protecting against pairwise bugs.

Dramatically reducing the number of tests.

Pairwise bugs represent the majority of combinatoric bugs.

Pairwise bugs are more likely to happen.

Test case creation tools are available.

The authors go on to explain many examples of pairwise testing. The paper chronicles lengthy academic studies culminating with the co-author's (Schroeder) study. His study concluded there is "...*no significant difference in the detection removal efficiency of pairwise test sets and same sized randomly selected test sets.*"

Pairwise testing fails when:

You don't select the right values.

You don't have a good enough oracle.

Highly probable combinations get too little attention.

You don't know how the variables interact.

Conclusion. Pairwise testing has permeated the testing world. It has invaded our conferences, journals, and testing courses. The pairwise story is one all testers would like to believe: A small cleverly constructed set of test cases can do the work of thousands or millions of test cases. As appealing as the story is, it is not going to be true in many situations. Practitioners must realize that pairwise testing does not protect them from all faults caused by the interaction of 1 or 2 fields. Practitioners must realize that blindly applying pairwise testing may increase the risk profile of the project.

Encouraging SME Companies to Adopt Reviews in Their Entirety

Ilkka Tervonen, Ph.D

Lasse Harjumaa

Dept. of Information Processing Science

University of Oulu,

Oulun yliopisto, Finland

Dr. Ilkka Tervonen is a professor of software engineering at the University of Oulu. His current research interests include software quality, software inspection, and process improvement. He heads the research team i3GO (Improved Inspection Initiative Group in Oulu). He received his doctorate in software engineering from the University of Oulu in 1994.

Lasse Harjumaa is a lecturer and doctoral student at the University of Oulu. His doctoral focuses on inspection process improvement by means of process patterns. His current research interests include software inspection, agile methods, and process improvement.

Small and medium-sized enterprises (SME) in the field of software have recognized quality as an asset, but they tend to have rather a limited view of possible techniques. They are familiar with different types of testing but are often lacking in knowledge of software reviews and inspections. This paper introduces the results of a long-term research program conducted with SME software companies.

This research project began informally in 1998 with a learning process revealing that "a fool with a tool is still a fool." The sense is that introducing an inspection tool within a company that does not have an established development process will likely fail.

From 2001 till the present, the research studied six companies, four of them SME, and two similar sized nationwide companies. The study focused on three approaches:

1. Tools support for Web-based inspection
2. Review process improvement
3. Motivation factors and obstacles

Tools. Many tools were described built by University of Arizona, Lucent, Adobe, University of Oulu, and Microsoft. Approximately 50 students used each of three tools in a distributed environment to inspect a requirements specification. The benefits of the tools that were most appreciated were adaptability to a variety of platforms and the attachment to a Web browser. The collaborative aspects of inspection are facilitated by Web technology because the Web is global and work is not limited to working hours. This not only introduces flexibility into the inspection meeting but it also enables easy, manageable distribution of the artifacts of the inspection. However, the job of moderator can be burdensome in this environment.

Review process improvement. The research found that established frameworks for improvement (CMM, BOOTSTRAP, SPICE, TMM, Test Maturity Model, TPI, Test Process Improvement) were not sufficiently focused on inspection. Accordingly they invented their own model. The model rated the maturity of 12 activities associated with inspections by measuring the implementation of 29 indicators associated with those activities. Implementation was graded on a scale of 1-5.

Na = not relevant

0 = not in use

1 = partially in use, exists to some degree

2 = largely, exists fairly well

3 = fully, always in existence

Maturities of the activities in the six companies ranged from 22-67 % based on a weighted rating formula.

[Editor comment: Regrettably no conclusion was drawn as to what these percentages mean as compared with more recognized models like the CMMI.]

Motivations and obstacles. An interview was conducted in the spring of 2004 in 12 enterprises ranging between 160 and 5 people.

The interviewees were asked to rate the importance of the following motivation factors for performing reviews:

Sharing expertise

Sharing information

Teaching novices

Controlling project progress

Process improvement

Finding more defects

Finding defects earlier

Finding defects faster

The three "finding defects" categories outscored all other significantly with Sharing and Teaching a modest second choice.

The interviewees were also asked to rate the obstacles to reviews:

- Shortage of time
- Shortage of staff
- Not required
- Costs
- Lack of expertise
- Laborious
- Geography
- No need

Interestingly the most important obstacles were shortage of time and staff, while few felt that Need was not a valid or important obstacle. In conclusion the authors assert that achieving improvement is a challenging task that requires:

1. Support from management
2. A quality group keen on improvements
3. Allocation of adequate resources

Panel Discussion—XP/Agile Methods

Jean Richardson

Jean Richardson was the moderator for this panel. She has been working in hardware and software development environments since 1990. During that time she has led a number of process improvement initiatives, mentored individuals entering the field of technical and business communication, and led professional development and education efforts for technical communicators.

Wayne Allen

Wayne Allen is the co-founder of the Consultants Guild, an association of senior business and IT consultants. He began his professional IT career in 1987 where he has been involved in both small and large solutions for the public and private sectors. As an early adopter of Extreme Programming and Agile methods, he has been pushing the original boundaries of XP by meeting the real needs of the business world in the trenches. The core of his approach is “Your software development process must support the people and goals of the project and organization.” He has helped define realistic customer roles as well as integrating QA, UI design, database administration, and project management into Agile teams, all with a focus on the practical over the theoretical. Allen maintains a blog at <http://weblogs.asp.net/wallen/> and can be reached at wayne@ConsultantsGuild.com.

Ward Cunningham

Ward Cunningham is an architect in Microsoft’s Platform Architecture Guidance group where he focuses on social aspects of technical knowledge. He is a founder of Cunningham & Cunningham, Inc., has served as director of R&D at Wyatt Software and as principal engineer in the Tektronix Computer Research Laboratory. He is well known for his contributions to the developing practice of object-oriented programming, the variation called Extreme Programming, and the communities supported by his WikiWikiWeb. Cunningham hosts the AgileManifesto.org. He is a founder of the Hillside Group and has served as the first program chair of the Pattern Languages of Programs conference which it sponsors. He created the CRC-Card design method which helps teams find objects.

Sean DeMartino

Sean DeMartino has been developing software for more than nine years, with experience in the agricultural, banking, and manufacturing industries. The last five years he has been a software engineer for Intel Corporation. He is currently working in the Fabrication Sort Manufacturing (FSM) group. He has worked with several software development life cycle methodologies, primarily utilized the eXtreme Programming process for the past three years. His primary roles are as software designer/developer and eXtreme Programming (XP) coach. In these roles, he is responsible for mentoring and coaching teams on the XP software development process and designing/developing capacity analysis software. This software is utilized by Intel’s fabrication plant industrial engineers for strategic and tactical planning.

Susan Muse

With more than 11 years of experience in developing software, Susan Muse has played just about every role on a software development team. She is currently managing the technical implementation of ITIL at Intel. She initially turned to XP out of sheer desperation when she realized that her project was as likely to succeed using waterfall as a penguin learning to fly. Over the past year, she overcame the traditional doubts associated with change to gain developer, customer, and management support for this business value focused software development methodology. Muse has been instrumental in mapping XP practices to CMMi and supporting other Intel groups and projects embarking on XP.

Fei Xie

Fei Xie received his doctorate in computer science from the University of Texas at Austin under the supervision of Professor James C. Browne, and joined the faculty at the Department of Computer Science, Portland State University, in 2004. His research interests are primarily in software engineering, especially software safety, security, and reliability. He is particularly interested in developing techniques and tools that are based on formal methods and enable design and development of safe, secure, and reliable software systems. He is also interested in hardware/software co-design and co-verification.

This panel provided an opportunity for the audience to question XP/agile methods practitioners about any aspect of the methodologies. The panel outlined the major characteristics and features of XP:

- Short iterations
- Frequent feedback
- Test before code
- Programmer pairs
- Customer on team
- Tester as part of the solution
- Refactoring

The audience had many challenging questions including:

- How do you make the customer available?
- Is the customer empowered to accept on-the-spot changes?
- Are programming pairs logistically practical?
- How do you get by without documentation? When are you done?

(cont. on p. 12)

SYNOPSIS CONTINUED

How good is the software when you get done?

The answers were all positive as expected from a pro XP/agile panel. The audience didn't appear to know enough about it to form an enlightened opinion or ask tougher questions. Only about one-fourth of the audience responded affirmatively when the panel asked how many are familiar with XP/agile methods. When asked how many were skeptical or opposed to XP/agile, I was the only one to raise my hand. (One other told me later that he hesitated but didn't know what to ask to challenge the panel.) When asked why I doubted XP, I asked the panel if there were any data, other than anecdotal, to substantiate the post-delivery quality level of the product or that the process save measurably in time and dollars over a traditionally developed product. Naturally I was told yes, but no data source was offered. I was told that NASA was using the technique in a clandestine environment, with no data available. It was generally acknowledged that the methodology is not likely to catch on soon in the regulated world, certainly not for safety critical applications.

Mike Kress is an associate technical fellow for the Boeing Commercial Airplane Group in Seattle. He is a Senior member of ASQ and immediate past chair of ASQ's Software Division.

FDA COMPLIANCE CORNER

BY DAVID WALKER

This column provides a concise update each quarter on computer validation subject matter. The goal of this column is to provide valuable information to Software Division members employed in FDA-regulated industries such as medical devices, laboratories, clinical research and development, and manufacturing of food, drugs, and cosmetics.

On September 17, 2004, a citizens petition was delivered to the U.S. Food and Drug Administration. The petition requests that the FDA revoke 21 CFR Part 11 in its entirety.

Furthermore, it urges FDA to "pursue implementation of electronic record and signature systems through the application of Government Paperwork Elimination Act (GPEA) standards and the enforcement of predicate rules."

The 16-page formal petition was submitted by the Industry Coalition on 21 CFR Part 11: http://www.chpa-info.org/web/advocacy/submissions/09_17_04_Coalition_Cit_Petition.pdf

The coalition believes that recently passed federal laws should be used to support this rather than the prescriptive regulation.

Great Sites:

FDA Center for Radiological Health Searchable Recognized Consensus Standards:

<http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfStandards/search.cfm>

NIST Information Technology Laboratory, Computer Security stuff:

<http://www.itl.nist.gov/>

Validation of computers, methods, and data in analytical laboratories:

<http://www.labcompliance.com/index.htm>

Please send me more great stuff to share in this column; send to David.Walker@Stryker.com.

SQE & ISO CONSULTANT

CSQE Online CLASS*



\$495 per person

- Course also available onsite
- ASQ section taught with a 95% pass rate
- Certified as a CQManager, CQA, CQE, CRE, CSQE, & RAB-LA
- Perform gap analysis & internal audits

*Based on ASQ BOK

ROBIN L. DUDASH
INNOVATIVE QUALITY PRODUCTS & SYSTEMS, INC.

phone/fax 724-789-7424
iqps@aol.com • www.iqps.net

FROM THE REGIONS

Region 4 Chris FitzGibbon

This forum provides Software Division members with information on software quality events in Region 4 (Canada).

Western Canada

The **Software Quality Assurance Vancouver User Group (VanQ)** provides Vancouver-area software practitioners the opportunity to share knowledge and experiences. In October, the VanQ presentation was on the ASQ awards process with an emphasis on recognizing individuals and organizations that promote the establishment of software best practices. Other recent guest speakers have delivered impressive presentations on software development and testing tools and techniques. VanQ meetings are held at the Burnaby campus of the British Columbia Institute of Technology. More information is available from the group's Web site: <http://www.vanq.org>.

The Calgary-based **IEEE/ASQ Discussion Group for Software Quality** also has some impressive presentation topics, including "The Role of the Q/A Manager" and "Mixing AGILE and Traditional Approaches." The group meets every two weeks at the Calgary campus of the DeVry Institute. All sessions are free and advance registration is not required. Their Web site is <http://www.software-quality.ab.ca>.

If you have information that you would like to share with fellow ASQ Software Division members, or you would like to get more actively involved with the division, contact me at chris@orioncanada.com or 613-563-9000. It would be a pleasure to hear from you.

Region 5 Joel Glazer

The Baltimore Section had a Software System Safety presentation as a dinner topic this winter. There are other activities in Region 5—DC, North Virginia, New Jersey, and Philadelphia SPIN meetings are continually taking place and drawing respectable sized turnouts. SSQ of DC is also drawing participation—these activities supplement ASQ meetings in the region. What is lacking is a sufficient number of candidates to hold CSQE refresher courses through local community colleges. If anyone has suggestions as to how best to increase the number of CSQE candidates, please provide you inputs to the regional councilor.

Region 6 Tom Gilchrist

The Pacific Northwest Quality Conference (PNSQC.org) held its 22nd

conference October 11-13 in Portland, OR. (Mike Kress provides a full report on the conference elsewhere in this newsletter.) They still have fond memories of the joint conference that ASQ Software Division participated in in the late 1990s. This year's conference features more than 40 educational sessions in four focused tracks: metrics, management, testing, and process. For more information, visit <http://www.pnsgc.org>

We are excited about the ASQ World Congress on Quality and Improvement (WCQI 2005) being held in Seattle in the spring of 2005. The Software Division will have some track items of interest to software practitioners. The Seattle Area Software Quality Assurance Group has been helping the division make this a special event.

If you're in the Seattle area on the third Thursday of every month (except December), SASQAG holds monthly public meetings in the Seattle area at Attachmate in Factoria. SASQAG also supports certification and study groups. If you are in the area and want to attend, please look at <http://www.sasqag.org> for upcoming events, directions, and meeting time.

If you have information on local software quality and testing events in your area of Region 6, please send them to me for our events calendar. I am looking for more information about activities and events in California. Visit <http://www.tomgtomg.com/asq6> for information on events around Region 6.

Tom Gilchrist, Region 6 ASQ Software Division, tomg@tomgtomg.com.

Region 10 Nancy Poma

Hot Software Quality Topics—The 6th Annual Quality Conference in Michigan is already being planned for next October, and we need your ideas on hot topics and even hotter speakers for the Software Quality Track. There are also opportunities to help plan this event, which entitles you to free admission and credit toward maintaining your CSQE. Please contact nmpoma@comcast.net if you have ideas to suggest or if you are interested in helping with this event.

CSQE Refresher Sessions—The ASQ Greater Detroit Section 1000 offers a CSQE Refresher Course, with the next session scheduled to start February 26, 2005. The course fee is \$500. More information is available at <http://www.asqdetroit.org>.

GL-SPIN Monthly Meetings—The GL-SPIN meetings alternate between University of Michigan-Dearborn and Oakland University, at 7 p.m. on the second Thursday of each month. More information on topics and speakers is available at <http://www.gl-spin.org>.

Region 12 Irv Segal

I'm pleased to announce that thanks to efforts of several volunteers our first regional conference was a very big success. Sixty professionals attended and several have requested to sign up as new ASQ members. Our topic of "Leveraging SQA to Do More With Less" seemed to be a hit and the wrap-up discussion about "CMM Boom or Bust" led to a lively discussion.

We're already starting the planning for next year's conference and looking forward to building on our success. We're also getting involved with the local SPIN group and investigating what collaboration can be done there.

Irv Segal, SysGen, Inc., 847-205-5349, irv.segal@sysgeninc.com.

Region 13 Granville Jones

Granville Jones will teach a new Six Sigma in IT Project Management course at the University of Denver's University College. <http://www.universitycollege.edu>. University College is the master's-level school at DU. The Computer Information Systems course is the first of what will eventually be a complete six-sigma program at DU. The winter quarter 2005 course started January 3, and ends February 4. The weekly three-credit course will consist of lectures, practical exercises, and a six sigma case study project. For more information contact Granville at granville-jones@comcast.net.

Mark Your Calendars for the 2005 Rocky Mountain Quality Conference, April 18 & 19, 2005

The theme for this year's conference is: "QUALITY DRIVES THE BOTTOM LINE—Tools & Attitudes." A partial list of possible topics includes: economics of quality; quality systems/standards/ISO series; process mapping/improvement; Six Sigma; implementing organizational change; customer satisfaction; leadership; statistical tools; design of experiments; root cause analysis/corrective & preventive action; Lean thinking; manufacturing/service; project management.

More information about the conference is available at <http://www.rmqc.org>.

MARK YOUR CALENDARS FOR THE 14TH INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY (ICSQ)

This year's Software Quality Conference provides a forum for individuals and organizations seeking technologies, concepts, and techniques to improve the quality of their software products, processes, and services, as well as networking and learning opportunities. Attend any of the offered 24 sessions including a special Aerospace Software Quality Panel addressing "The Future of Software Quality."

14ICSQ Overview

- Sunday—Certification Exams
- Monday—Tutorials/Courses
- Tuesday—Conference Sessions
- Wednesday—Conference Sessions
- Thursday—Courses
- Friday—Courses

Full program available at
www.asq.org/softwareforum/

Where: The Wyndham Orlando Resort
in Orlando, Florida

When: Conference, March 22-23, 2005

To Register:

- Fax or mail your registration form in this edition of *Software Quality*, or
- Log on to www.asq.org/softwareforum/ or
- Call 800-248-1946.

Keynote Speakers

Mark Paulk

"Best Practices Sourcing"

Linda Westfall

"Communicating Our Way to Better Software"

Conference Sessions

Session A

Project Risk Management

James A. Ward

Session B

Measurement Adventure

Carol Dekkers

Session C

Risk Based Software Testing T

Theresa Hunt

Session D

Comparing the CMM, CMMI, and Related ISO Standards for Improvement and Measurement

Mike Kress

Session E

Software Quality and Change Management

Frank Voehl

Session F

Methods Panel

Scott Duncan

Session G

Cost of Quality and Return on Security

Taz Daughtrey

Session H

Systems Engineering and Systems Management Minimum Process Set for Mission Success

Mike Dimario

Session I

It's Still the Requirements

James Ward

Session J

Evolution of Software Quality Assurance

Michelle Pierce

Session K

Tricks for Designing Test Cases

Karen Bishop-Stone

Session L (3:00 p.m.-5:00 p.m.)

Aerospace Panel—The Future of Software Quality

With Mike Kress, Michelle Pierce,
Mark Paulk, Dave Zubrow, Joe
Jarzombek, Martha Wetherholt

Session M

Management's Role in Achieving Predictable Software Development

Steve Rakitin

Session N

Our Path to a Quality Management System

Sue Carroll-McGrath

Session O

Requirements and Business Rules

Karen Bishop-Stone

Session P

Identifying Risks to System-of- Systems Integration Efforts

Dave Zubrow

Session Q

Six Sigma Software

Richard Biehl

Session R

10 Things Every Software Tester Should Know

Mike Kress

Session S

Good Enough Software

Scott Duncan

Session T

Educating Software Quality Professionals

Trudy Howles

Session U

Exhibitor Presentation

The Westfall Team

Session V

Achieving CMMI Level 4

Doug Hamilton

Session W

Traceability in Data Transaction Algorithms

Evelyn Richardson

Session X

Modeling CMMI Process Improvements

Tony Timbol

www.asq.org/softwareforum/



14th International Conference on Software Quality

Preconference Tutorials—March 21

Conference—March 22–23

Post-conference Courses—March 24–25

Orlando, FL

REGISTRATION FORM

ASQ member # _____ First Name for Badge _____

Name (Last) _____ (First) _____ (MI) _____

Company Name _____

Title _____

Primary Mailing Address _____ (Circle one: Home or Business)

City _____ State/Province _____

Zip/Postal Code _____ Country _____

Phone _____ Fax _____

E-mail _____

Guest? (First) _____ (Last) _____

Concurrent Sessions

Please circle the sessions you wish to attend. (See p. 14 and conference Web site for list of sessions.) Although you are not bound by these choices, this will provide an expected group size to conference organizers. You will receive a confirmation letter of your registration via the U.S. Postal Service.

Tuesday, March 22, 2005

10:30 am – 11:30 am

Sessions: A B C D

1:30 pm – 2:30 pm

Sessions: E F G H

3:00 pm – 4:00 pm

Sessions: I J K L

Wednesday, March 23, 2005

10:30 am – 11:30 am

Sessions: M N O P

1:30 pm – 2:30 pm

Sessions: Q R S T

3:00 pm – 4:00 pm

Sessions: U V W X

CONFERENCE REGISTRATION RATES

Two-Day Registration

\$695.00 - ASQ Member/SW Division Member

\$795.00 - Nonmember

One-Day Registration

\$395.00 - ASQ Member/SW Division Member

\$495.00 - Nonmember

Full-Day Tutorials – March 21, 2005

(TUT01) Cost of Quality and Return on Security Investment – Taz Daughtrey
\$395.00 ASQ Member / \$495.00 Nonmember

(TUT02) Software Testing Techniques – Theresa Hunt
\$395.00 ASQ Member / \$495.00 Nonmember

(TUT03) Software Document Preparation – Karen Bishop-Stone
\$395.00 ASQ Member / \$495.00 Nonmember

(TUT04) Software Quality Standards: Why and What – Scott Duncan
\$395.00 ASQ Member / \$495.00 Nonmember

(TUT05) Basic Statistics for Software Engineers – Mark Paulk
\$395.00 ASQ Member / \$495.00 Nonmember

(TUT06) Function Point Measurement—Demystified – Carol Dekkers
\$395.00 ASQ Member / \$495.00 Nonmember

(TUT07) Software Requirements Engineering – Linda Westfall
\$395.00 ASQ Member / \$495.00 Nonmember

Half-Day Tutorials – March 21, 2005

(TUT08) 8:00 am – Noon – How to Define Short, Usable Processes – Tim Olson
___ \$245.00 - ASQ Member/SW Division Member
___ \$345.00 - Nonmember

(TUT09) 1:00 pm – 5:00 pm – Using a Measurement Framework to Successfully Achieve Measurable Results – Tim Olson
___ \$245.00 - ASQ Member/SW Division Member
___ \$345.00 - Nonmember

Post-Conference Courses – March 24-25, 2005

#05207C – Software Auditor Skills
___ ASQ Member/SW Division Member \$1095
___ Nonmember \$1195

#05208C – Software Configuration Management
___ ASQ Member/SW Division Member \$1095
___ Nonmember \$1195

PAYMENT INFORMATION

Purchase Order # _____

___ Check enclosed (make payable to ASQ) Check # _____

Please charge my credit card: ___ Amex ___ MC ___ Visa

Credit Card # _____

Expiration Date _____

Cardholder Name _____

Cardholder Address _____

Signature _____

To register call **800-248-1946** or **414-272-8575** or visit
<http://www.asq.org/softwareforum/conferences>

OFFICERS

Doug Hamilton, Chair
Accenture
312-693-0308
douglas.b.hamilton@accenture.com

David Walker, Chair-elect
Stryker Instruments
269-323-7700
david.walker@stryker.com

Mike Kress, Immediate Past Chair,
Nominating Chair
425-717-7038
michael.p.kress@boeing.com

Eva Freund, Treasurer
703-573-7466
Eva.Freund@nara.gov

Yvonne Kish, Secretary
469-441-6149
yvonne_kish@yahoo.com

Theresa Hunt, Vice Chair Programs
14ICSQ Chair
407-834-5825
theresahunt@earthlink.net

Karen Bishop-Stone, Vice Chair
Technology
952-925-1156
karen@testware-assoc.com

Hank Sobah, Vice Chair Membership
Services
412-937-7687
hsobah@innovativesystems.net

CHAIRS & OTHER CONTACTS

Sue Carroll, Examining and
Awards Chair, Journal Editor
SAS
919-677-8000, ext. 17032
Sue_Carroll@bellsouth.net

Evelyn Richardson, Publications
Chair
703-531-6589
evelyn.richardson@ngc.com

Robin Dudash, Education Chair
iqps@aol.com

Linda Westfall, Bylaws Chair
The Westfall Team
lwestfall@westfallteam.com

Alex Hilgendorf, Certification Chair
Unity Health Insurance
608-273-4679
alex.hilgendorf@unityhealth.com

Taz Daughtrey, Liaison Chair
540-568-2778
daughtt@jmu.edu

Patricia McQuaid, World Congress
California Polytechnic State
University
805-756-5381
pmcquaid@calpoly.edu

Carol Dekkers, World Conference
on Quality and Improvement
Track Chair
Quality Plus Technologies, Inc.
727-393-6048
dekkers@qualityplustech.com

Rufus Turpin, Marketing Chair
Carpe Diem Infomatics, Inc.
613-715-9146
rufus@carpedieminfo.ca

Scott Duncan, Standards Chair
706-649-2345
softqual@knology.net

REGIONAL COUNCILORS

Region 1—Eric Patel
RapidSQA
877-749-4586
epatel@rapidsqa.com

Region 2—Jean Burns
Universal Instruments
607-779-7868
burns@uic.com

Region 3—Ruth Pennoyer
NY/NJ Metropolitan Section
973-325-7592
pennoyer@ruthpennoyer.com

Region 4—Chris FitzGibbon
Orion Canada, Inc.
613-563-9000
chris@cyberus.com

Region 5—Joel Glazer
Northrop Grumman ES
410-765-2346
joel_glazer@md.northgrum.com

Region 6—Tom Gilchrist
The Boeing Company
425-234-4865
tgilchrist9@attbi.com

Region 7—James Hutchins
Boeing
714-762-0712
james_h_hutchins@yahoo.com

Region 8—Michael S. Kiefel
Abbott Laboratories
614-624-7973
Michael.kiefel@abbott.com

Region 9—Lee Black
Irwin Mortgage Company
317-537-3361
lblack@insightbb.com

Region 10—Nancy Poma
EDS
248-265-0456
nmpoma@comcast.net

Region 11—Robert Galen
EMC Corporation
919-248-6392
galen_bob@emc.com

Region 12—Irv Segal
SysGen, Inc.
847-205-5349
Irv.segal@sysgeninc.com

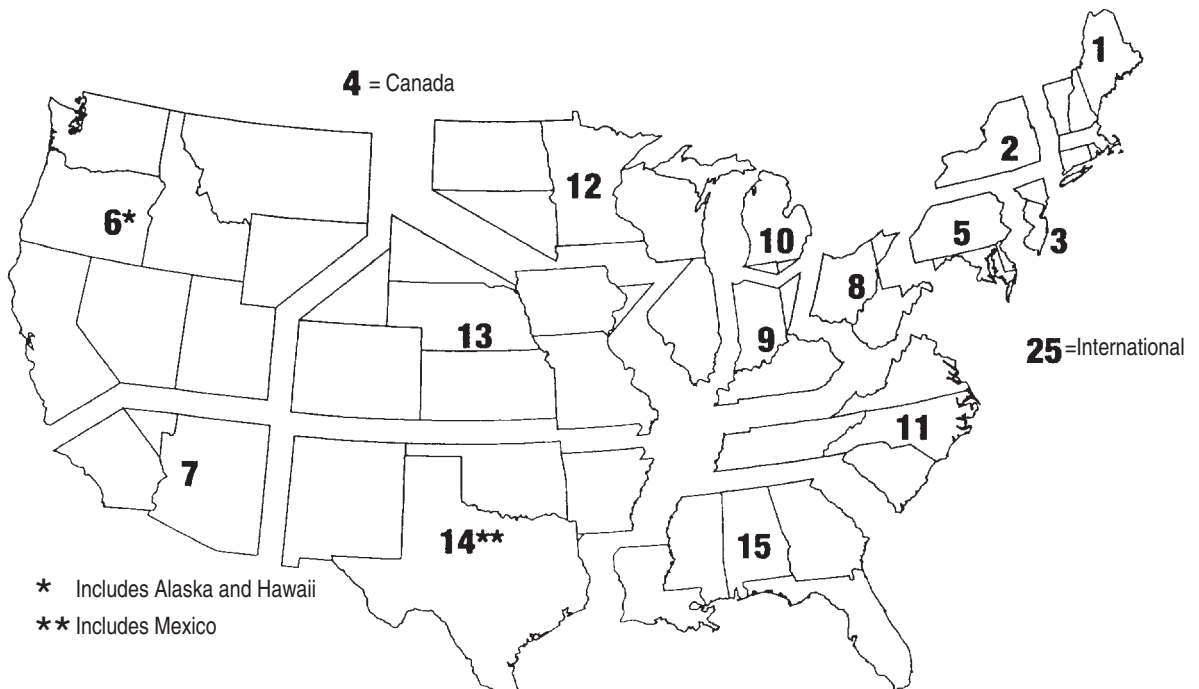
Region 13—Granville Jones
JVJ Enterprises
303-969-0228
granville.jones@att.net

Region 14—W. L. 'Bill' Trest
Lockheed Martin Aeronautics
Company
817-777-7598
bill.l.trest@lmco.com

Region 15—Mark Neal
Alcon Laboratories, Inc.
407-384-1659
mark.neal@alconlabs.com

Region 25-International—
Zigmund Bluvband
A. L. D. Ltd
+972-3088-5100
+972-5496-7201 cellular
4+972-3977-5095
zigmund@ald.co.il
www.aldservice.com

Regional Map



SOFTWARE QUALITY

SUBMIT ARTICLES FOR THE NEXT ISSUE OF *SOFTWARE QUALITY* BY MARCH 11, 2005.

EVELYN V. RICHARDSON
PHONE: 703-531-6589 • FAX: 202-628-0599
E-MAIL: EVELYN.RICHARDSON@NGC.COM

DON'T FORGET TO MARK YOUR CALENDARS

CSQE EXAM

Location	Exam Date	Application Deadline
Orlando, FL, at 14ICSQ	March 20, 2005	February 25, 2005
ASQ Local Sections and International Sites	June 4, 2005	April 1, 2005

EDITOR

EVELYN V. RICHARDSON
Northrop Grumman Information Technology
Federal Enterprise Solutions
voice: 703-531-6589
fax: 202-628-0599
e-mail: evelyn.richardson@ngc.com

EDITORIAL REVIEW BOARD

DOUG HAMILTON, Chair
DAVID WALKER, Chair-Elect
EVELYN RICHARDSON, Publications Chair

EDITORIAL POLICY

Unless otherwise stated, bylined articles, editorial commentary, and product and service descriptions reflect the author's or firm's opinion. Inclusion in *Software Quality* does not constitute endorsement by ASQ or the Software Division.

ADVERTISING

FULL PAGE-\$500 per issue
1/2 PAGE-\$250
1/4 PAGE-\$125

Yes! Please enter my subscription to *Software Quality Professional*, a quarterly publication focusing on the needs of professionals in software quality.

Member Number _____

Subscriber information—please send to:

Name _____

Company Name/Agency _____

Title _____

Address _____ Apt. /Suite # _____

City _____ State/Province _____

Zip+4/Postal Code _____ Country _____

Telephone _____ Fax _____

E-mail _____

Payment options:

All orders must be paid in U. S. currency. Please make checks payable to ASQ. Checks and money orders must be drawn on a U. S. financial institution. All prices are subject to change without notice.

Payment enclosed: Check Money Order Amt. Paid _____

Please charge: Visa MasterCard American Express

Charge Card No. _____ Exp. Date _____

Cardholder Name (please print) _____

Signature _____

Cardholder Address _____

City _____ State/Province _____

Zip+4/Postal Code _____ Country _____

Telephone _____ Fax _____

Subscribe by:

Phone 800-248-1946 or 414-272-8575 (outside North America)

Fax 414-272-1734

Mail ASQ Customer Care Center, P. O. Box 3005, Milwaukee, WI 53201-3005

Online: <http://sqp.asq.org>

SOFTWARE QUALITY PROFESSIONAL			
	ASQ Members	Nonmembers	Institutional
U. S.	\$45.00	\$75.00	\$120.00
International	\$70.00	\$95.00	\$150.00
Canada	\$65.00	\$95.00	\$150.00

Software Quality Professional is published in December, March, June, and September. Subscription is for one year.

Priority Code: QRSADD1