A BZ Media Publication

Software Test & Performance

VOLUME 3 • ISSUE 7 • JULY 2006 • $8.95

www.stpmag.com

BEST PRACTICES: Defect Tracking

How Your Test Plans Can Benefit From Mind Mapping

Improve Quality Through Better Risk Identification And Analysis

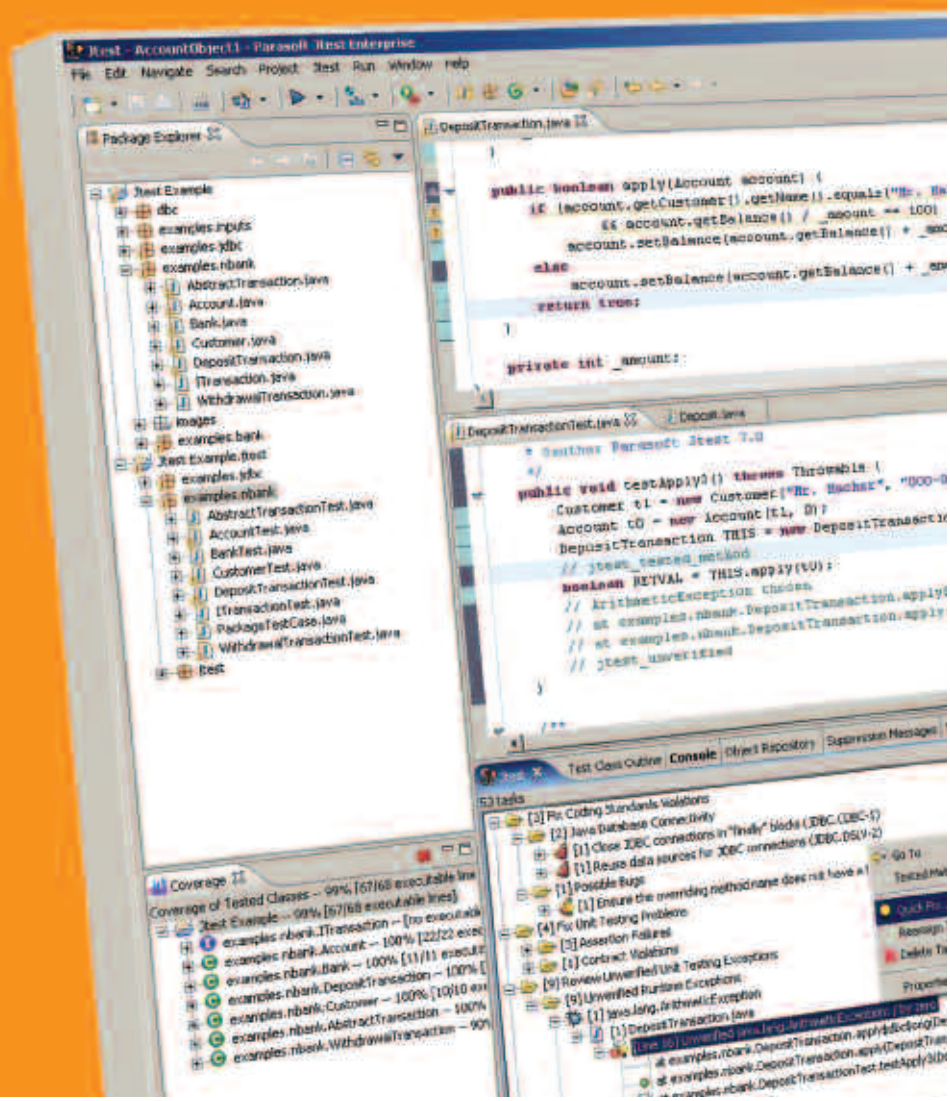
Write On With Ruby

This Open-Source Scripting Tool Can Help Automate Web Services Testing

Ferret Out Functionality Problems With Test-Case Execution State Histories

Code in quality. Test out bugs.

Take control of your code through Automated Unit Testing and Code Analysis…



PARASOFT®

Jtest®

- Automatically analyzes your code, applying 500+ Java coding best practices that surface poor code constructs affecting reliability, security and performance.
- Automatically unit tests your Java code evaluating code behavior and exposing unexpected exceptions, boundary condition errors and memory leaks.
- Generates extendable, high coverage test cases in JUnit format that can be quickly turned into libraries of reusable test assets.
- 100's of Quick Fix options for fast, accurate resolution of errors.
- Integrated code coverage analysis ensures that your code is thoroughly tested.
- Team collaboration support allows you to define, distribute and enforce code quality standards across entire development teams.

Stop bugs before they start:

Jtest automates the valuable, yet often tedious tasks of code reviews and unit testing allowing development teams to adopt a "test as you go" strategy that promotes testing code as it is developed so that quality is built into the code from its inception and bugs eliminated before they infect the rest of the code base.

The result? Cleaner, safer, more reliable and consistent code. Reduced code rework. Faster, more predictable release cycles. Reliable applications and happier, more productive end-users. **To try Jtest today, call 888-305-0041 (x-3501) or go to www.parasoft.com/STPmag.**

Parasoft Jtest clients include: Lehman Bros., Cisco, Disney, Raytheon, ADP, Sabre Holdings, GE and more.

PARASOFT®
We make software work.™

Automated Error Prevention™

SAVE Over $300! Register by July 14 For Super Early Bird Savings!

The industry's independent conference for enterprise software developers and IT development managers using Eclipse tools and technologies

eclipse WORLD
The Enterprise Development Conference

Hyatt Regency Cambridge
Boston, MA
September 6-8

Save Over $300!
Register by July 14 for
Super Early Bird Savings!

At EclipseWorld, you will...

- Learn how to save money and improve developer productivity with Eclipse.
- Improve team collaboration using Eclipse-based tool sets.
- Go beyond the IDE to understand the wide range of Eclipse technologies.
- Get deep inside Eclipse's open-source architecture.
- Discover the best, most effective Eclipse add-ins and plug-ins.
- Learn how to build better applications using the Eclipse RCP.

Are you ready for Eclipse?

Whether you're an Eclipse master, just getting started with the platform, or trying to decide if Eclipse technologies are right for your development team, the EclipseWorld Conference & Exhibition is the #1 educational event that you should attend this year.

"The Eclipse Foundation is proud to participate in the second annual EclipseWorld conference in September 2006. The debut 2005 EclipseWorld in New York was an excellent educational event for enterprise IT professionals. We're glad that the Eclipse community has embraced this independent technical conference."
Mike Milinkovich, Executive Director, Eclipse Foundation

"Well organized and enjoyed the presentations."
Alice Chan, Software Engineer, CyberAccess

"Good sessions on the basics as well as plug-ins."
Patricia Timms
Senior Software Engineer, Kodak

"It's a really good place to get those questions answered which are hard to find in books."
Farha Azaz, Senior Software Engineer, TouchNet Information Systems

"A good opportunity to meet Eclipse developers of all kinds."
Suzanne Yoakum-Stover, Senior Computational Scientist, SAIC

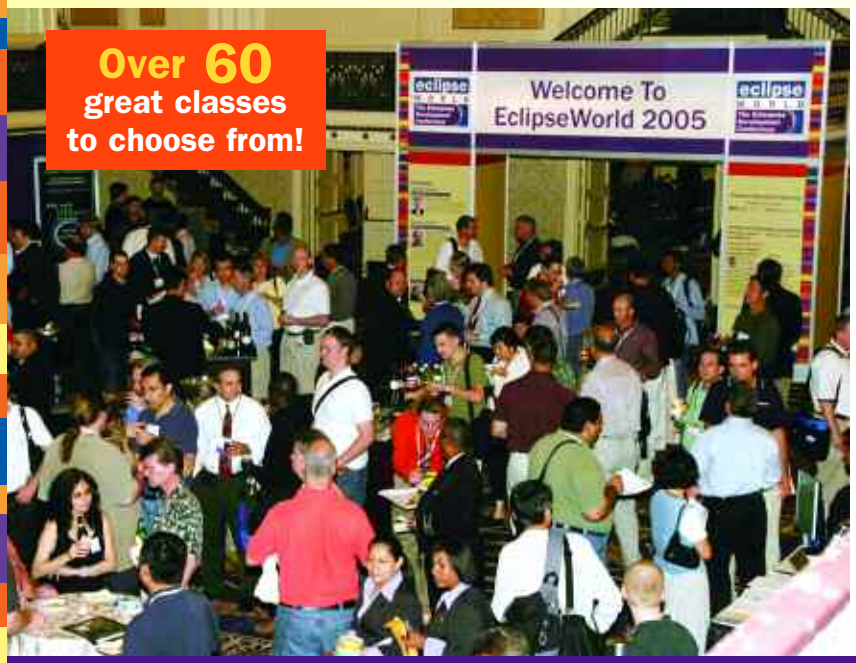
"If you are looking for both high and low level information regarding Eclipse, both as an IDE as well as an RCP, you'll find it at EclipseWorld. The presenters are knowledgeable and approachable. In addition, the networking with other users of the Eclipse platform is invaluable."
Dan Colbert, Product Manager, CompassCom

"EclipseWorld was dynamic, thought provoking and cutting-edge."
Robert Rothman, Senior Developer, Morgan Stanley

"EclipseWorld had a great collaborative atmosphere. It was exciting to learn about the different uses of Eclipse, and see what people were creating with it."
Justin Stern, Software Engineer, Auspice

"Technology being discussed in the conference was very leading edge. I got to familiarize myself with projects I hadn't had a chance to research."
James Pitts, Director of Database Programming, Embarcadero Technologies

Over 60 great classes to choose from!



Keynote Speaker

Mike Milinkovich is the executive director of the Eclipse Foundation. In the past, he has held key management positions with Oracle, WebGain, The Object People and Object Technology International Inc. (which subsequently became a wholly owned subsidiary of IBM), assuming responsibility for development, product management, marketing, strategic planning, finance and business development.


Mike Milinkovich
Executive Director
Eclipse Foundation

A BZ Media Event

Diamond Sponsor
COMPUWARE
www.compuware.com

Platinum Sponsor
SYBASE

Gold Sponsor
openmake
a product of catalyst

Media Sponsors
SD Times
Software Test & Performance
Eclipse Review
EclipseSource
acm queue
eclipse magazine
BZ Software

BZ Media is an Associate Member of the Eclipse Foundation.

For sponsorship or exhibiting information, contact Donna Esposito at 415-785-3419 or desposito@bzmedia.com.

www.eclipseworld.net

Get More

Get 110% satisfaction with TestTrack Pro.

It's not often you feel 110% satisfied about a product. But, TestTrack Pro gives you more advanced issue management features, more performance, and more ease of use than any other solution on the market today.

- **More issue management** with defect tracking and change request capabilities, configurable workflows, extensive field customizations, and comprehensive reporting
- **More performance** with fast and secure remote access, scalable cross-platform client/server support, full Unicode support, and seamless integration with popular IDEs and SCM tools
- **More ease of use** with easy installation and administration, and uncomplicated licensing options

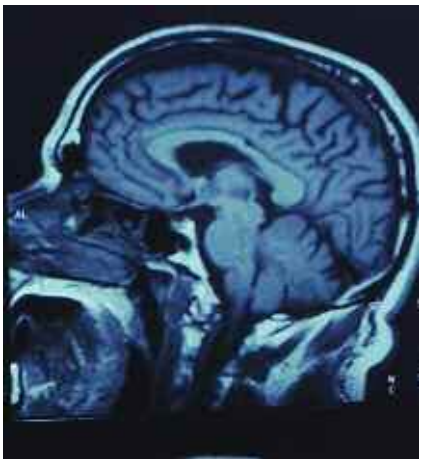
But wait... there's even more. With every Seapine solution, you get world-class support before and after the sale. In addition, every product is backed by our unconditional 30-day, money-back guarantee. Now it's your turn to get more. Download your **FREE** fully functional evaluation software now at **www.seapine.com/st&p** or call **1.888.683.6456**.

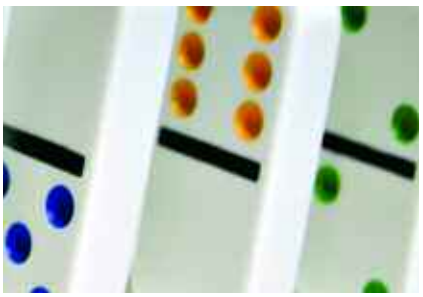
Seapine Software™
managing process, change, and quality throughout the enterprise

Software Test & Performance
VOLUME 3 • ISSUE 7 • JULY 2006

Contents

A BZ Media Publication





Departments

CodePro AnalytiX™ is developed by the experts who brought you the popular book
Eclipse: Building Commercial Quality Plugins — Eric Clayberg & Dan Rubel

Unfair Advantage

tools that make every developer a quality expert

CodePro AnalytiX™

for Eclipse, Rational® and WebSphere®

Ready for IBM Rational

eclipse READY™

Download a risk-free trial copy:
www.instantiations.com/codepro

Drive out quality problems earlier in the development process.

Powerful code audits and metrics make every developer a Java expert.

Create tight, maintainable systems.

Wizards generate unit and regression tests, and analyze testing code coverage.

Enforce quality measures across teams.

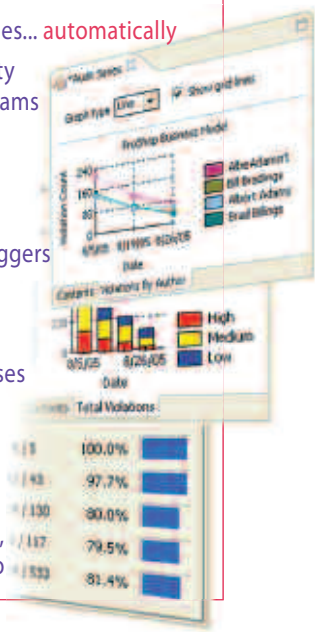
Define, distribute, and enforce corporate coding standards and quality measures, across individuals and teams… no matter where they're located.

Spend less time and money to develop high-performance Java systems.

Dramatically improve your application development process and code quality… leading to more reliable, maintainable systems.

Key Features of CodePro AnalytiX™

- Detect & correct code quality issues... automatically
- Define, distribute & enforce quality standards across development teams
- 800+ audit rules and metrics, 350+ Quick Fixes
- Powerful management reporting
- Code metrics with drilldown & triggers
- Audit Java, JSP and XML files
- JUnit test case generation
- Code coverage analysis of test cases
- Dependency analysis & reporting
- Integrated team collaboration
- Javadoc analysis & repair
- Seamless integration with Eclipse, Rational® and WebSphere® Studio



instantiations
the development tools experts

www.instantiations.com 1-800-808-3737

No one outside of IBM has more experience creating Eclipse-based Java development tools

eclipse FOUNDATION MEMBER™

IBM Business Partner™

Software Test & Performance
VOLUME 3 • ISSUE 7 • JULY 2006

Software Test & Performance (ISSN #1548-3460, USPS #78) is published 12 times a year by BZ Media LLC, 7 High Street, Suite 407, Huntington, NY 11743. Periodicals privileges pending at Huntington, NY and additional offices.

POSTMASTER: Send address changes to BZ Media, 7 High Street, Suite 407, Huntington, NY 11743. Ride along is included.

The Right Tool For the Job

Lindsey Vereen
Editor

During the first season of "Saturday Night Live" (you may have seen it on reruns), cast member Chevy Chase delivered the fake news each week. This was in 1975, the year that Francisco Franco, the infamous Spanish dictator, died—a fact that Chevy Chase never failed to mention on his newscast ("After 11 weeks, Generalissimo Francisco Franco is still seriously dead.").

What do you suppose Mr. Chase would be saying about Microsoft Windows Vista right about now—that it's still seriously late?

While you can't test quality into a product, thorough testing is critical for the huge code base that makes up Windows Vista. Like its predecessors, Vista will have to support a broad range of legacy hardware and software. And testers have a formidable challenge to tackle: Ship dates continue to stretch, and pieces of the delivered product have a habit of falling off the table.

In this tricky situation, tool quality must remain paramount. In the reviews written for technical specialists in hardware and software design and test periodicals, tools can seem arcane and even intimidating. Some are difficult to use, and few come shrink-wrapped. Often they're designed to work with very specialized products. Readers would love to have objective reviews of the tools they use, but writing said reviews is no easy task—it's tough even to come up with benchmarks to use.

Enter "The Decider"

So how do you decide which tools are the keepers and which ones should be left on the shelf? How do you get an idea of a product's market acceptance? Here's one way: Let the tool users themselves decide which products they like or dislike—and we'll publish the results.

Although tools are important, the person wielding the tool is the motive force behind it; that person's job is made easier by the tool. So who better to evaluate tools than the people whose careers depend on them?

Which brings us to the announcement of the second annual Software Test & Performance Testers Choice Awards. These awards recognize excellence in software test and performance management tools, encompassing the broad range of tools designed to improve software quality.

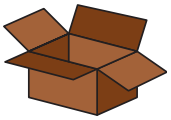
And the Winner Is...

Here's how the process works. Beginning this month, test and performance management tool vendors may nominate any or all of their tools for consideration by you, the readers of this magazine.

In September, subscribers to Software Test & Performance will have the opportunity to vote for your favorite, most useful tools. You'll receive an e-mail from us disclosing a unique link to the electronic polling place, so keep a close eye on your inboxes. And, remember, you can vote only once.

We'll announce the winners at the Software Test & Performance Conference in November. Those of you who can't make the trip to Boston for the conference will get the scoop in the December issue of Software Test & Performance. You can find all the details about the contenders and the process itself at www.stpmag.com/testerschoice.

Information about terrific tools is definitely no replacement for thorough, impartial product evaluations, but it may help you forestall complaints that your next product is still seriously late. ☒

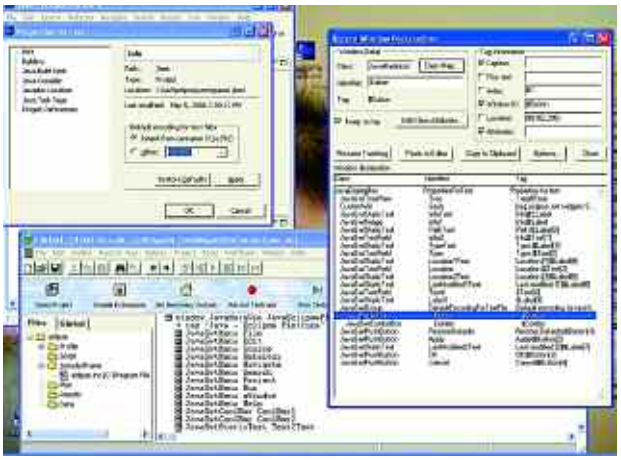


Out of the Box

Compiled by Alex Handy

Borland Releases SilkTest 8.0

Though Segue Software put in long hours to build seven prior versions of the SilkTest functional testing tool, Borland has taken the credit for version 8.0. This edition was basically complete when Segue was acquired by Borland earlier this year, but the changes made to the software fall right in line with Borland's recently professed affection for Eclipse.



SilkTest's new home in Eclipse offers testers more flexibility when working with applications based on the open-source IDE.

As such, SilkTest 8.0 brings its capabilities directly into the Eclipse IDE. That means that SilkTest 8.0 can now perform its automated functional tests on applications based on Eclipse 3.0 and 3.1.

"SilkTest provides a sophisticated, multiplatform system for automating functional testing—a critical component when taking a life-cycle quality management approach to software delivery," said Erik Frieberg, vice president of product marketing at Borland. "SilkTest 8.0 enables users to realize the strengths of SilkTest for their applications developed in Eclipse and combines with the full family of Silk products to validate and improve performance while managing the testing process from one central point."

SilkTest 8.0 is available now from Borland at www.borland.com. The software costs US$6,500 per seat and runs on Windows.

DriverPacks.net Makes Building QA Systems Easy

Two years ago, a geek known improbably enough as Bâshrat the Sneaky decided to try something many thought was impossible: He wanted to collect all of the device drivers for Windows XP and put them on a single CD. Now, two years later, he has finally completed his project. DriverPacks.net offers free CD images that can be downloaded and used to install drivers for sound cards, graphics cards, wireless adapters, motherboard chipsets and CPUs. These CDs are absolutely indispensable for lab administrators struggling to keep all those regression systems up-to-date.

"I've started these DriverPacks because I wanted to achieve a Uniform UXPCD," wrote Bâshrat the Sneaky on DriverPacks.net. "Why in the world would you want support for all available hardware? I hear you thinking, 'Well, that's easy to explain: to be able to use these unattended Windows XP installation CDs on any computer (of course, one that's capable of running Windows XP).' So in that thread at MSFN.org, I asked how I could do that. Of course people were laughing at me (a quote: 'I think you'd be naïve to think you'd be able to fit the (latest) drivers of every device made since XP was launched, on the one CD'). After a while I decided to just try it, and I started the DriverPack Sound. It was hell to get it fully working, but the positive feedback kept me continuing."

Bâshrat the Sneaky's driver packs are available for free on the Web site. Send him an e-mail to thank him if you enjoy his work.

QA Wizard 4.0 Teleports to Users

The latest version of QA Wizard from Seapine Software (www.seapine.com) brings automated functional testing into multiple environments and platforms.

QA Wizard is designed to automate the functional and regression testing of Web-based and Windows applications. It's an intelligent object-based tool, according to the company, in that it knows the difference between an input field and a static text object. The goal of QA Wizard is to allow users to focus on what they want to test instead of how to test it. "QA Wizard 4.0 includes Firefox browser playback support, detailed script reports, .NET 2.0 control support and additional status tool functionality," said Jay Luis, director of marketing for Seapine. "This release also includes improved performance of Java application startup procedures, increased speed and accuracy when searching for Java controls and variable data support for checkpoints in Java test scripts," Luis added.

Seapine also released the latest version of its source code management tool in March. Surround SCM integrates with QA Wizard 4.0, as well as with other Seapine tools, said Richard Riccetti, president and CEO of Seapine.

"With its enhanced Java support, QA Wizard 4.0 can now be used to automate testing for most Windows, Web and Java applications. Surround SCM 4.1 includes additional IDE support, making it easy for users to work within the environment of their choice," Riccetti added. "When combined with TestTrack Pro 7.5, Seapine's cross-platform issue-tracking solution, Surround SCM and QA Wizard provide an unbeatable end-to-end application lifecycle management solution."

QA Wizard 4.0 is priced near US$4,000 per seat and runs on Windows. A 30-day trial version can be downloaded from the company's Web site.

Wily's Introscope 7 Offers a Deep Look

Computer Associates' Wily Technology Division (www.wilytech.com) has completed work on Introscope 7, the latest iteration of the company's flagship Web application monitoring tool. When installed, Introscope gives users a deep look under the tablecloth of a Web services deployment. From inside the Introscope console, administrators can monitor and report bottlenecks, slowdowns and errors encountered as their applications go about their daily business.

"Control over Web applications is essential for both IT and business success," said Dick Williams, senior vice president and general manager of CA's Wily Technology Division. "With Introscope 7, organizations can manage and optimize the largest applications across the most complex environments—including service-oriented architectures and heterogeneous infrastructures. By collaborating with our customers, we have put more of our application management expertise 'in the box' by adding advanced analytics and automatic baselining features in addition to significant usability enhancements."

New to this version is customer-centric, end-to-end management capabilities, which allow Introscope to observe every aspect of a user's interaction with the monitored application. The software is available now with pricing information directly from CA.

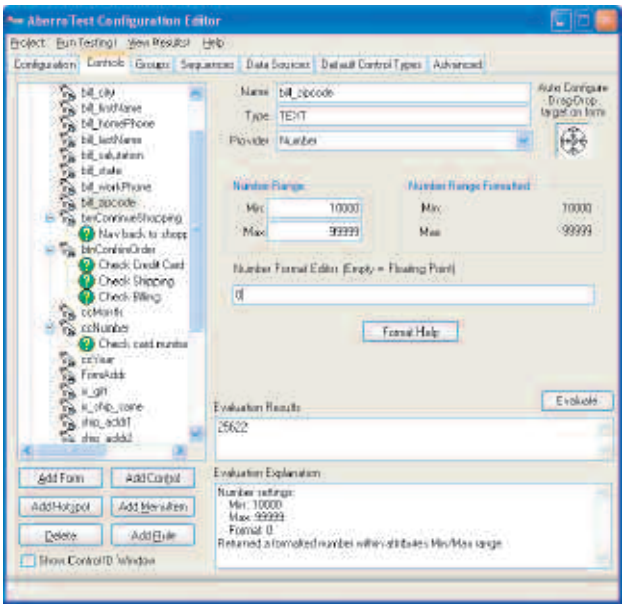
AberroTest Finds Its Way

Functional testing doesn't have to mean sitting a tester down in front of a monitor with a giant list and a two-liter bottle of soda, according to neophyte testing company Aberro Software (www.aberrosoftware.com). The Maryland-based company released a new functional test tool called AberroTest in May. The tool is designed to randomly generate its own paths through GUIs and then save all the information gleaned from these excursions.

AberroTest requires a bit of initial setup to perform its duties, however. The tool must be shown the way around an interface and buttons, and text boxes and toggle switches must be marked for identification purposes. Once this is done, however, testers lay out a goal for the software to accomplish, such as writing a value to the database or saving a file externally. At that time, AberroTest attempts to push its way from the default starting screen towards its goal by randomly clicking, poking and typing information into the application's many interface items. When it accomplishes its goal, it starts over and tries a new path toward success. Thus, from some simple initial conditions, thousands of random tests are generated and run.

AberroTest also catalogs these successes and failures for future reference. Users are given a full report of the functional capabilities of their applications upon completion of each test run. The tool itself was developed over the course of three years of university research, and Aberro, which formed in November, was created specifically to put this research to use in a commercial functional testing tool.

AberroTest is available directly from the company. The tool costs US$3,999 per seat and runs in Windows.



Pointing out the interface controls for an application must be done before an application can be tested with AberroTest.

Compuware Ships New Version of Its Load Tester

As the load-testing tool for Java, .NET and Web applications, QALoad 5.5 from Compuware (www.compuware.com), offers a new way to manage test rules. QALoad's parameterization wizard gives developers a point-and-click interface to create rules for testing applications under heavy loads. These rules can then be saved and used later to build the robust testing conditions necessary to feel out an application's strengths before deployment.

"Knowing how an application will perform in production is crucial to business success. Without the proper focus on preproduction performance assurance, poor service levels can result in lost productivity, negative perceptions of the business and losses to the bottom line," said Elizabeth Maly, director of Compuware Quality Solutions. "Preproduction performance testing with Compuware QALoad lets businesses consistently deploy and deliver high-quality applications with confidence."

QALoad 5.5 costs around US$50,000 for 100 virtual users simulated and runs under Windows.

Send product announcements to stpnews@bzmedia.com

An Evolution In Performance Testing


Scott Barber

I recently returned from WOPR6, the sixth meeting of the Workshop On Performance and Reliability (www.performance-workshop.org). An ongoing series of invitation-only, minimal-cost peer workshops for experienced performance testers and related professionals, WOPR builds skills in system performance and reliability, and allows people who are interested in these topics to network with their peers. The emphasis is on mutual learning, sharing hands-on experiences and solving practical problems. In these semiannual meetings, groups of approximately 20 subject-matter experts and experienced working practitioners discuss pertinent state-of-the-art topics with the intent of advancing the state-of-the-practice of performance and reliability testing through open collaboration and cross-pollination of ideas.

WOPR6 was organized to explore the concept of evolving perceptions of performance testing. We accomplished this through reports of relevant experiences from past projects or current initiatives, which demonstrate or contradict the view that performance testing is currently undergoing a period of significant, rapid and positive change.

As you might imagine, participants' perceptions of the field's status varied widely. During the run-up to the workshop, participants expressed positions ranging from "Things are getting worse!" to "Same stuff, different day" to "Performance testing is advancing so quickly, I can hardly keep up!"

This time, Google was kind enough to serve as host to the workshop at the Googleplex in Mountain View, just south of San Francisco. For six straight days, approximately 40 individuals from around the world with a passion for performance testing met and discussed personal experiences that, to them, represented performance testing as stagnating, advancing, evolving, poising for a paradigm shift or something else altogether.

At the Tipping Point

Entering the workshop, I had a sense that in some ways, the state-of-the-art of performance testing was nearing a conceptual and technological tipping point likely to cause a dramatic improvement in how we test systems. However, I sensed that many of these new advancements had not yet permeated the industry as a whole, a feeling that six days with these topics did not dispel.

That isn't to say that significant advancements aren't happening—they are. But they're apparently occurring in small, often isolated pockets. It seems that for every one organization that implements one of these advancements, there are 10 more that consider, but don't implement the same advancements. It appears to me that many companies may be trapped by inertia.

Although it would be neither possible nor productive to summarize the six days of facilitated and passionate conversations among experts and experienced practitioners, I'll share some of the key points that I gleaned. Specifically, I'd like to discuss opinions that either corroborate or oppose some positions I've put forth in this column and elsewhere.

Evolution Isn't Always Good

Brian Warren, a manager of performance testers at a company that is frequently listed in Fortune magazine's top 1,000 companies, shared an experience about how advancements in performance testing in his organization have resulted in an increasing amount of data to process and analyze. On the surface, this appears to be good, since we've been complaining for years about not being able to collect enough of the right data fast enough to properly assess system performance. The challenge? Now that we have the data, we don't always know what to do with it. To illustrate what made him suspicious of some of the findings from all of this data, Warren related a story from his days as a geographer specializing in mapping.

It seems that the field of geographic mapping made some reasonably embarrassing observations based on the tons of new data available from global positioning satellites. With so much of this new satellite data streaming in so fast, to simplify processing, the first step was to "pixelate" or average the data into sections measuring 10 meters by 10 meters.

While this degree of granularity was great for macro-level mapping, averaging over 33 square feet means that things like most houses, streams, drainage ditches and country roads simply disappear. I'm sure this is acceptable for, say, a general topographic map for the state of Colorado, but if the goal is to determine runoff patterns in Florida, you simply aren't going to get an accurate picture with

Scott Barber is the CTO of PerfTestPlus. His specialty is context-driven performance testing and analysis for distributed multi-user systems. Contact him at sbarber@perftestplus.com.

data granularity of 10m x 10m. Warren shared that it took quite some time to realize that some of their applications of the data were simply not accurate and potentially misleading without "de-pixelating" prior to analysis.

I took two important lessons away from Warren's experience. First, I'm reminded that evolution is a set of slow and more or less naturally occurring changes typically brought on by an alteration in environment. Sometimes these evolutionary changes are critical for survival, but at other times, they cause more harm than good. Warren's story reminded me not to get too excited about seeing evolution until I'm sure that the evolution is going some place positive. In this case, the simple existence of more data does not automatically solve many problems.

Second, I'm reminded of just how messy our data often is, and how easy it can be to compile a few averages to make that data easier to understand. The problem is that those averaged averages can easily pixelate some of our best indicators of poor performance right out of sight. Even worse is the fact that most of the tools we use start by presenting us with pixelated data, which makes it that much more tempting to start our analysis from there instead of remembering that that data is a summary and going instead to the raw data before we plunge into our detailed analysis.

Prototyping Performance

In 2001, I presented a paper at the Pacific Northwest Software Quality Conference that included an experiment I'd been working on: I had folks sit in front of a Web site I had created and rate their satisfaction with the time it took each page to download. Using a little JavaScript, I'd made it so that different pages took different amounts of time to display.

The exercise was designed to help me determine a user's tolerance for delay using qualitative methods that I could then directly convert to quantitative data. Mostly this data was used to show the writer of the performance requirements that, for example, an eight-second download time simply wasn't acceptable to the users of this application, no matter what research they quote to the contrary.

I have to admit, until performance testing consultant Roland Stens shared with me his experience using prototypes to collect performance requirements, I had become rather convinced that my little idea made a cool paper, but was pretty much hopeless in industry due to a dearth of prototyping. Stens, however, seems to have found a solution to the prototyping problem—his team uses Axure RP (www.axure.com), which is an affordable and usable prototyping tool. While Axure RP doesn't come with a "How long would you like this page to take to load?" GUI button, it took Stens all of six lines of JavaScript to include this functionality! In my view, that's more than a little exciting.

Oh, did I mention that Stens was completely unaware of my paper and that his approach was very well received by his client?

Robinson's next brainteaser was a killer: 'If you use a machine gun, you don't have to aim that carefully.'

Paradigm Shift?

Harry Robinson, an expert in model-based testing, spoke to us about his minimal but significant performance testing experience, making some wonderfully quotable statements. First, he remarked that when doing performance testing, you're more likely to find what you're looking for by hunting "for pessimal behavior as opposed to optimal." While this is probably not a new perspective for the folks reading this column, I think that Robinson stated it particularly well.

He continued by making the point that "domain experts can be dangerous." We have all seen that most job posts for performance testers list domain expertise near the top, yet even a performance testing novice realizes this is often a mistake. The fact is that while domain knowledge can improve your test design, it can also tend to create a situation called "inattentional blindness," which occurs when we don't think to test or pay attention to a situation that could turn out to be extremely relevant because, for example, "no *real* user would ever do *that*."

Robinson's next brainteaser was, "If you use a machine gun, you don't have to aim that carefully." In other words, if you have the ability to test lots of different combinations and scenarios, it may make more sense to just start testing them instead of spending a bunch of time building a model that has a low probability of being correct anyway.

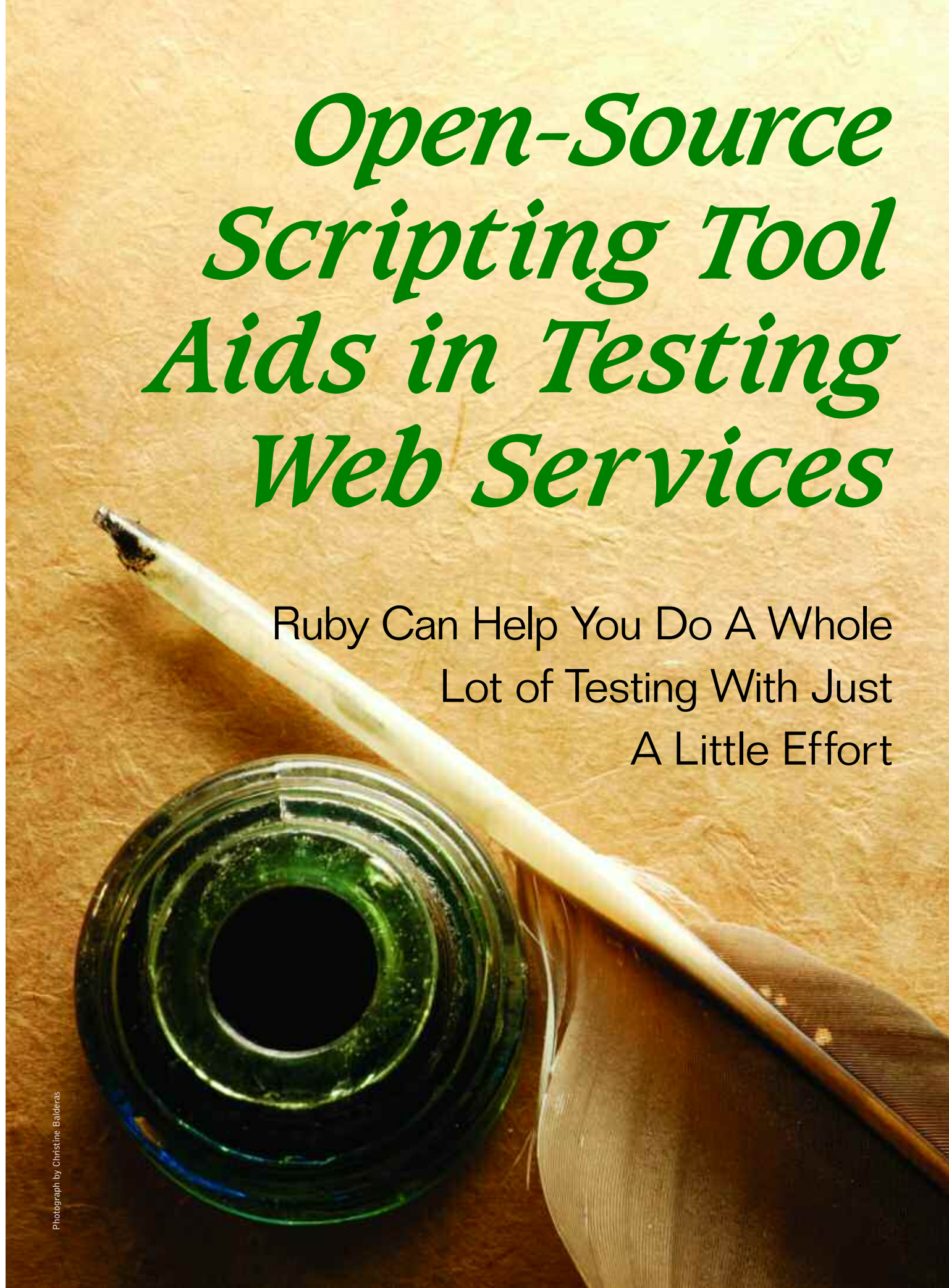
Finally, after presenting his experience and listening to many of ours, Robinson said, "I don't know if you're experiencing a paradigm shift in how you model performance usage, but you should be." While that's a bigger topic for another day, I will say that I felt more than a little bit validated that the person I have come to know as "The Model Guy" came to the same conclusion as I, along with others, had when faced with the challenge of trying to figure out how to design our performance tests to get the most important or meaningful results efficiently.

It seems that we've reached the bottom of our second page together once again, and I have many more key points from the workshop to share with you. So I'm going to do to you what every one of the few television shows I catch occasionally have done to me this week: get you all interested in what is coming next, only to say...

... To Be Continued ☒

Open-Source Scripting Tool Aids in Testing Web Services

Ruby Can Help You Do A Whole Lot of Testing With Just A Little Effort



Photograph by Christine Balderas

By Michael Kelly

Many systems developed today involve the use of Web services. I will assume you know what Web services are, as well as some of the basic concepts behind them, but you may not know what makes testing Web services unique. The toughest challenge in testing Web services is the absence of a user interface, which often makes it difficult for testers who typically test applications only manually. This means that some level of automation, or the use of specialized tools, is necessary to perform the testing. There are many good tools that can help, but nothing beats creating your tests with a scripting language.

Testing Web services is exciting to me because it requires an intriguing mix of black-box testing skills and coding skills. It also adapts well to automation (no cumbersome GUIs getting in the way) and still requires you to think about the underlying business logic behind the service. In this article, I'll share some of the scripting I've done on a recent financial services project with the open-source, object-oriented programming language called *Ruby*. I used Ruby to create test XML files based on production data, submitted those files to the Web service I was testing, and parsed the resulting XML for validation. I was able to execute thousands of different XML test cases in only a few days of work.

The Ruby code in this article is simple and straightforward, but it's also powerful. It allowed me to do a lot of testing with very little effort. I'll start with some simple code (connecting to the Web service) and end with some more simple code (validating the response I get back). I'll add my disclaimer now: I'm not a developer. This is Ruby code written by a tester for a tester. I make no claim to the fastest or most elegant code—it just works.

We'll walk through the Ruby code a step at a time. Looking at code in this manner can sometimes be distracting for some people. Often, it helps to see all the code at once. For those of you who would like to see the sample code in its entirety, you can download the Ruby script at www.stpmag.com/downloads/stp-0607_galen.zip.

Using Ruby to Connect to A WSDL Service

The first thing you need to do is establish a connection to your Web service and get some data moving back and forth between your script and the service. Ruby offers many ways to do this, none of which is particularly well documented. In the end, I had to team up with a developer to get the final code working, but when all was said and done, it took us only a couple of days, part-time, to get it figured out.

The first service we connected to was a WSDL (Web Service Description Language) service. We used the Ruby Simple Object Access Protocol (SOAP) libraries to make our call. The basic logic goes as follows:

1. Take an XML file (for us, each XML file was a test case) and wrap it in a SOAP envelope. *Note: A SOAP envelope element is the root element of a SOAP message. It defines the XML document as a SOAP message.*
2. Create a connection to the WSDL service.
3. Send the SOAP envelope to the service (HTTP post).
4. Wait for a response.
5. Receive the response XML (or the error).

Following is the code we used to implement that logic. I'll walk you through it one section at a time, but if you're just getting started, Brian Marick has a better "first look" article on scripting with Ruby. I have a full reference to it below.

First, we need to include the various soap libraries we will be using:

```
require    'soap/element'
require    'soap/rpc/driver'
require    'soap/processor'
require    'soap/streamHandler'
require    'soap/property'
```

Next, we declare a constant for the endpoint (the *endpoint* is the location of the Web service) and we assign our XML to a variable:

```
LOCALHOST_ENDPOINT = "http://local-
host:8080/services/Service"
request_xml_string = 'xml...'
```

In the code snippet above, request_xml_string is assigned the XML request that the Web service is expecting. The next line creates a connection to the service using HTTPStream Handler:

```
stream = SOAP::HTTPStreamHandler.new
(SOAP::Property.new)
```

Once we've defined the connection, it's time to build the SOAP envelope, which is made up of a header and a body. The following creates the header and the body, using the service method getResponse and the XML stored in the variable above:

```
header = SOAP::SOAPHeader.new
body_item =
SOAP::SOAPElement.new('getResponse',
request_xml_string)
body = SOAP::SOAPBody.new(body_item)
envelope = SOAP::SOAPEnvelope.new(header, body)
```

Now we're ready to create the actual request string using the envelope and send it to the service. Once it's sent, we wait for the response to be returned and store the response value in the variable resp_data:

```
request_string = SOAP::Processor.marshal(envelope)
request =
SOAP::StreamHandler::ConnectionData.new(request
_string)
resp_data = stream.send(LOCALHOST_ENDPOINT,
request, 'getResponse')
```

The above code creates the XML string from the Ruby SOAP objects, sends the string and stores the response. I know I glossed over some of that, but that's because I got some help from a real developer. There are several different ways to do this, and this is just one example. If, like me, you aren't a Ruby wizard, I recommend working with a developer to get your connection

Michael Kelly is an independent consultant who focuses on test automation and exploratory testing. Kelly also serves as a director at large for the Association for Software Testing. You can reach him by e-mail at Mike@MichaelDKelly.com

code working. It's the only tricky part in this entire article.

Once we had a working script that allowed us to send our XML test cases, we were ready to start testing. Initially, we validated all the result XML manually—nothing beats the power of the brain—but over time, that work became tedious and not very fruitful. We then started building in automated validation for certain XML elements as they came back from the service. Once that was done, we had only to check for test case–specific information when validating results.

Using Ruby to Generate XML

We used Ruby and JUnit (see the *Using Ruby With JUnit* sidebar for more information) for all of our traditional black-box functional testing, but we wanted more. All of the Web services we were testing were new interfaces to existing financial services applications. That meant we needed to do a very large amount of regression testing for a very large amount of data. If you can't tell from the process discussed above, each planned test case we executed was a fair amount of work.

We needed a faster way to regression test; something that would allow us to generate a lot of test cases, run them, do some basic validation, and then do manual testing in areas where problems were identified. So we decided to start using Ruby to generate XML test cases based on existing production data. That process turned out to be simpler than we thought it would be, going pretty much like this:

1. Open the file containing all the policy data that will be used to generate the XML files.
2. For each line in the file, create a new XML file with that policy data.
3. Close the file.

We could do this because the service we were testing was a simple query service. It would be more difficult if the service were more complex.

The XML we generated followed the ACORD standard (insurance.xml .org/standards), and all the data in the file dump was very basic policy data: policy number, contract number and version number. With those three values we could generate our request for the service. With the following code, we were able to generate around 500 test cases every second (depending on the size of the data dump we sent it).

We used Ruby and JUnit for all of our traditional black-box testing, but we wanted more.

The first line in the code opens the file containing all the policy data dumped from production:

```
myDataFile = File.open('C:/Service Testing/Service A Testing/12.22.05 - policy data dump.txt', "r")
```

Then, for each line in the file (myDataFile) we want to create a test case file (or XML request):

```
myDataFile.each { | line |
    #Build the XML file...
}
```

The .each method executes the block of code following it for each line in myDataFile. The current line is stored in the line object.

Following the creation of all the test cases, we close the data file:

```
myDataFile.close()
```

The rest of the code in this section goes inside the each block (where we have the comment *Build the XML file…*). The first thing to do once we read in each line is split up the data for ease of use. In Ruby, we have the .split() method, which does just that. The following line of code parses the data wherever it finds a space between the data elements. It then stores that data in an array:

```
policyInfo = line.split(' ')
```

That code splits the policy information, storing the policy number in policyInfo[0], the policy version in policyInfo[1], and the contract number in policyInfo[2]. Next, we need to create the new XML file for each line. Something similar to the following should work for that:

```
newXMLFile = File.open('C:/Service Testing/Service A Testing/Request XML/' + policyInfo[0] + '-rq.xml', "a")
```

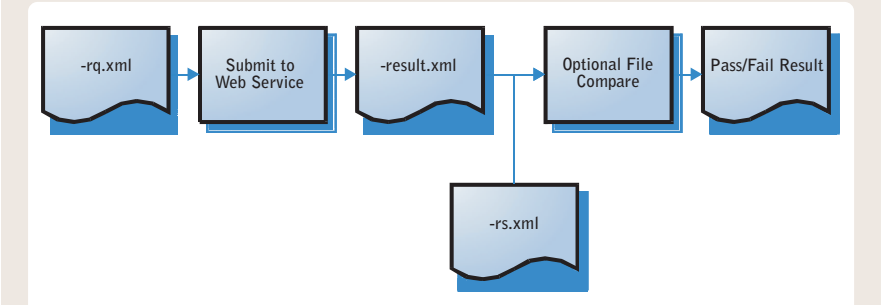
Notice that in the code above, we name the XML file according to the policy number. That's because the policy number is the only thing that's unique in the data. We also had an XML file-naming convention that included three possible extensions: –rq,–rs and –result.

The -rq file contains the request XML for the call to the Web service.

The -result file contains the actual result XML for the call to the Web service. This file can be compared to the expected result file (-rs) to see if the service returned the correct XML response. This file is automatically overwritten with every call to the Web service.

The -rs file contains the expected result XML for the call to the Web service. This file can be compared to the actual result file (-result) to see if the service returned the correct XML response. This file must be manually

FIG. 1: XML FILE-NAMING CONVENTION



LISTING 1

```
newXMLFile.puts('<?xml version="1.0" encoding="UTF-8"?>')
newXMLFile.puts('<ACORD
xmlns="http://www.ACORD.org/standards/PC_Surety/ACORD1.7.0/xml/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">')
newXMLFile.puts('      <SignonRq>')
newXMLFile.puts('                  <ClientDt>2005-12-22</ClientDt>')
newXMLFile.puts('                  <CustLangPref>en-US</CustLangPref>')
newXMLFile.puts('                  <ClientApp>')
newXMLFile.puts('                              <Org>Financial Services Company</Org>')
newXMLFile.puts('                              <Name>com.service.serviceA</Name>')
newXMLFile.puts('                              <Version>0.1</Version>')
newXMLFile.puts('                  </ClientApp>')
newXMLFile.puts('      </SignonRq>')
newXMLFile.puts('      <InsuranceSvcRq>')
newXMLFile.puts('                  <RqUID>00000000-0000-0000-0000-
000000000000</RqUID>')
newXMLFile.puts('                  <PolicySyncRq>')
newXMLFile.puts('                              <TransactionRequestDt>2005-12-
22T13:47:45.00000-05:00</TransactionRequestDt>')
newXMLFile.puts('                              <AsOfDt>2005-12-22</AsOfDt>')
newXMLFile.puts('                              <Producer>')
newXMLFile.puts('                                          <ProducerInfo>')
newXMLFile.puts('                                                      <ContractNumber>' +
policyInfo[2].chomp() + '</ContractNumber>')
newXMLFile.puts('                                          </ProducerInfo>')
newXMLFile.puts('                              </Producer>')
newXMLFile.puts('                              <PolicyNumber>' + policyInfo[0] +
'</PolicyNumber>')
newXMLFile.puts('                              <PolicyVersion>' + policyInfo[1] +
'</PolicyVersion>')
newXMLFile.puts('                  </PolicySyncRq>')
newXMLFile.puts('      </InsuranceSvcRq>')
newXMLFile.puts('</ACORD>')
```

overwritten or updated.

Figure 1, which depicts the XML file-naming convention and process for file use, shows how we used those files in our testing.

Building the XML file is actually a trivial exercise (or task) that amounts to little more than a bunch of puts commands for each line of the XML we want to write. Puts writes out a string to an I/O object (in this case, a File) and appends a new-line character if the string doesn't already end in one. We just copied and pasted the XML into our Ruby script, wrapped it with puts commands and inserted the few unique values we needed in the appropriate places. For example, look at Listing 1.

Finally, before we finish with the new XML file, we want to close it:

```
newXMLFile.close()
```

Executing the above code yields an XML request file for each policy in the data dump. Once we had the test data, all we needed to do was run it through the Web service using our earlier script. But that's not all we wanted to do. Since we're talking thousands of test cases with multiple points where each can fail, we also wanted simple validation and detailed logging of results. User-friendly logging is a must. We wrote our results to csv files for easy formatting and sorting.

Using Ruby to Validate XML and Log Results

Once we had our bazillion test cases, we needed to do the following:

1. Delete the old results file before running the test and create a new one (I could have just created a new one with a time-date stamp, but I took care of versioning a different way).
2. Get all the files in the directory and sort them.
3. For each file in the directory, execute the service call, check for errors, and validate the results that are returned.
4. Close the results file.

First, we want to delete the old results file and create a new one stored the path to the file in a variable to make the code a bit more readable:

```
bulkTestResults = 'C:/Service Testing/Service A
Testing/Test Results.csv'
if File.exists?(bulkTestResults) then
File.delete(bulkTestResults) end
myResultsFile = File.open(bulkTestResults, "a")
```

Next, we need to find all the files in the directory. I like to sort them by name so that my results are sorted by policy number (that's how I named them when writing the XML). In the following code, the glob method returns an array of file names matching the specified wildcard pattern (I use * just to capture all the files in the directory).

```
eachFileInDirectory = Dir.glob('C:/Service
Testing/Service A Testing/Request XML' + '/*').each
{ | file | file.downcase }.sort
```

For each file returned we change all the characters in the name to lowercase and then once all of them have been changed to lowercase we sort the array eachFileInDirectory. Changing the file names to all lowercase renders the results easier to read in the csv because it makes all the file names consistent.

Again, we'll need to use the each method. For each file in the directory, we want to process each test case file:

```
eachFileInDirectory.each do | testCaseFile |
        #Do something with the file...
end
```

At the end of the script, we need to close the results file:

```
myResultsFile.close()
```

Within the testCaseFile loop, we use a flag for test case failures. That way, we know to stop processing the other checks we perform on the response when a failure occurs. So the first thing we do when we enter the loop is reset that flag for the test case being executed:

```
testCaseFail = false
```

Next, we use the code covered above to send the request to the service and capture the response. The response then gets written out to the "–Result" file. That's done for debugging and archiving purposes. Next, we'll look at how to open the results file and validate the results.

In the code that follows, I'll show an example of validating only one type of element (the policy number). You can repeat that code as much as you like for any elements you'd like to validate.

They all look the same; just change the XML element and/or the value you're looking for. In this example, we're going to verify that the results we got back were the results for the correct policy number. If they aren't, then we'll log a message indicating that we didn't get the policy number we expected (or that we couldn't find the XML element containing the policy number).

The first thing we do is create a flag that indicates whether or not we found the XML element <PolicyNumber>:

```
tagFound = false
```

Then, we check to see if there was a failure in an earlier test validation check. If there was, we don't continue:

```
if not testCaseFail then
        #Rest of code here...
end
```

Inside that if statement, we open the results file for the test we just ran:

```
myTestCaseFile =
File.open(testCaseFile.chomp.split('\-')[0] +'-
results.xml')
```

Then we check each line in the results file for the <PolicyNumber> element:

```
myTestCaseFile.collect do | line |
        if line =~ /<PolicyNumber>/ then
                #More code here...
        end
end
```

In the code above, /<PolicyNumber>/ is a regular expression. If <PolicyNumber> is anywhere in the line, then =~ evaluates to true. If that regular expression evaluates to true, we'll want to check to make sure that the <PolicyNumber> element contains the correct policy number. If you remember from above, the name of the test case file is also the policy number. The next chunk of code uses the .include? method (which evaluates to a Boolean value) to check to see if the policy number is there. Notice that we perform some operations on the testCaseFile variable. All we're doing is isolating the actual file name from the full path to the file. We're stripping off both the path and the file extension. If the policy number isn't found, we log an error to our .csv results file and set the testCaseFail flag to true.

```
if not line.include?(testCaseFile.split('/')[-
1].chomp.split('\-')[0]) then
myResultsFile.puts(testCaseFile.split('/')[-1].chomp
+ ', FAIL, -result.xml, Could not find
<PolicyNumber> element that matched the policy
number requested.')
        testCaseFail = true
end
```

Whether we found the correct policy number value or not, before we exit our if statement we also need to set our tagFound flag to true. That way, we won't log an error indicating that we couldn't find the tag.

```
tagFound = true
```

Next, we want to close the test case results file:

```
myTestCaseFile.close()
```

And then we perform a check to see if we actually found the element we were looking for. If tagFound is false, we log an error and set testCaseFail to true:

```
if not tagFound then
myResultsFile.puts(testCaseFile.split('/')[-1].chomp
+ ', FAIL, -result.xml, Could not find
<PolicyNumber> element in response.')
        testCaseFail = true
end
```

We then repeat all that code for each verification we want to perform on the XML. I've thought about optimizing the code to make it more modular, but it works, and I currently perform a handful of checks on each file. At some point (once I have to start scrolling my script more than a couple of times) I'll rework the code to make it more modular.

That's all there is to it. These scripts made it possible to create several hundred request test case files in a few seconds and then execute them all and perform simple verifications on the results within about 45 minutes. All the results were logged in a .csv file that we could auto-filter based on result or error message.

This made it easy to identify all the affected policies when submitting defects. We were able to do all our traditional functional testing using hand-coded XMLs and submitting them using Ruby and JUnit, and then we did some high volume automation and found more than a handful of very obscure and hard-to-anticipate issues.

Scripting With Ruby Resources

As you begin your journey with Ruby, you can find advice, new ideas and friendly support at several rich resources.

First, check out a couple of articles by Brian Marick: "Behind the Screens" (www.testing.com/writings/behind-the-screens.pdf) and "Bypassing the GUI"(www.testing.com/writings/bypassing-the-gui.pdf). Marick is also working on a book titled "Scripting for Testers: Using Ruby" (Pragmatic Bookshelf, 2006), which will be a must-read for testers.

Another great place for Ruby tips and tricks for testers is www.Watir.com, which stands for Web Application Testing in Ruby. WATIR is an open-source test tool for automated testing with Ruby. They have a strong mailing list (which is searchable) and a lot of helpful people.

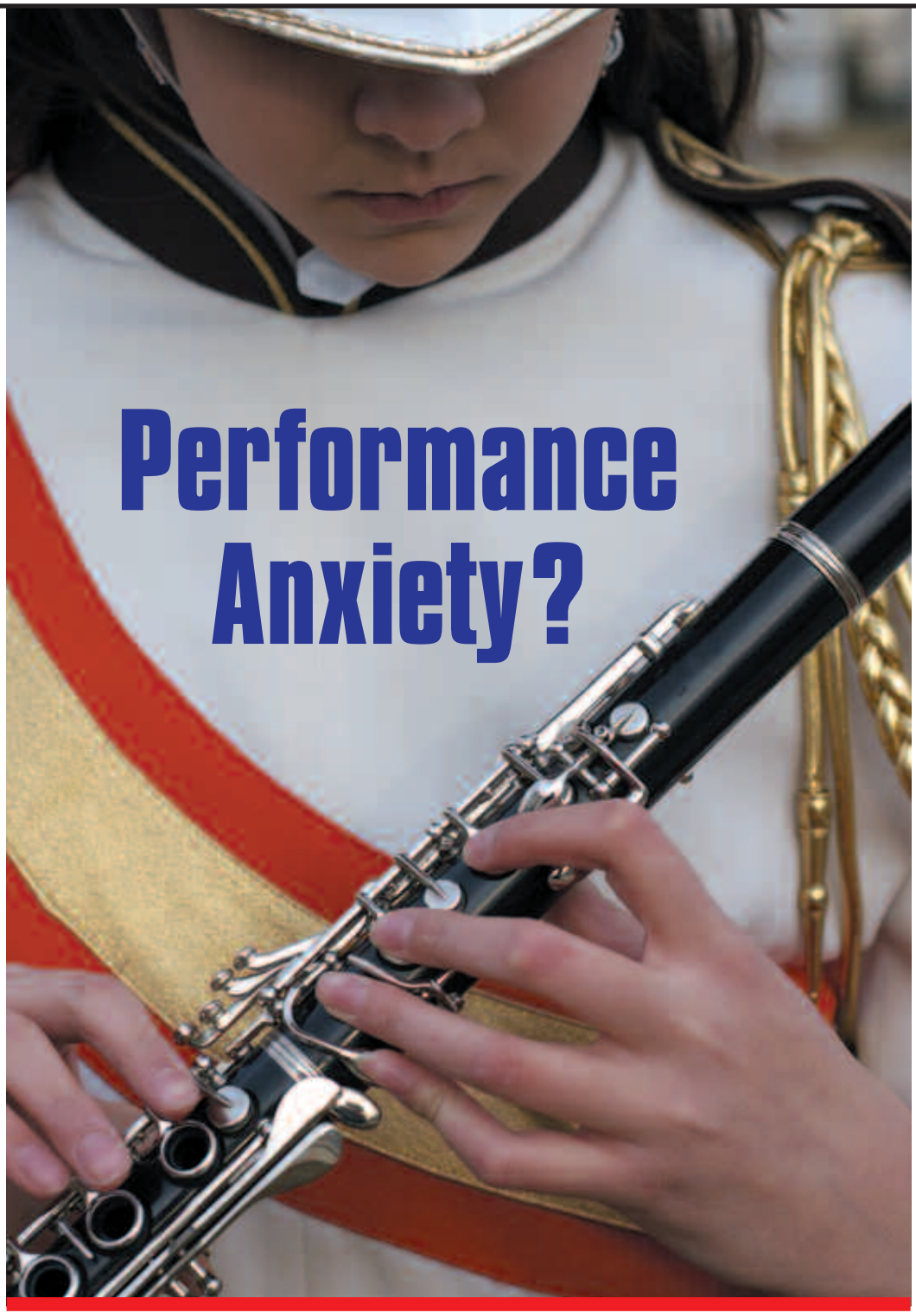
Finally, grab a copy of "Ruby in a Nutshell" by Yukihiro Matsumoto (O'Reilly Media, 2001). I've found it a must-have for writing Ruby code.

SMOKE TESTING WITH RUBY AND OUTLOOK

One little trick we picked up along the way was to have Outlook kick off our smoke test scripts (or build verification tests—depending on your preferred nomenclature). We were testing several services, and each of them ran in several different environments. So we wanted our smoke test scripts to be available to the project team for any service, in any environment, at any time. The problem with that is that only a couple of us had Ruby installed (it was not an "approved" technology at the time).

We needed a way for others to execute tests on our machines. Enter Outlook. All we needed to do was add an Outlook rule that would launch the correct script on the local machine based on the subject line of the e-mail. The requestor would send a headline of "Smoke Test" followed by the service name and environment, and Outlook would launch a batch file (I don't believe that Outlook can run Ruby scripts) that would launch the Ruby script. The test would execute and e-mail them the results.

It would also run in the background—so if I was working, I wouldn't even know it had happened (unless I was unusually observant that day). If you use the rubyw.exe instead of ruby.exe to execute your scripts from the batch file (rubyw.exe doesn't launch a DOS shell when it runs), you shouldn't even notice it running.



We can help... Software Test & Performance

Subscribe Online at www.stpmag.com

By Andrew Makar

Mind-Mapping for

How do you determine the scope of a testing strategy for software projects? Do you review the project charter, dive into a ream of requirement specifications, or start immediately with test case scenarios?

Whether the project team engages an independent quality-management team or uses an internal set of testing resources, determining the scope of the testing effort is a challenge best met early in the project life cycle.

Mature IT organizations depend on repeatable processes that apply a standard quality-management strategy, ensuring that quality is implemented throughout the project. But mature methodologies carry their own challenges: They can often produce lengthy test-strategy documents that are best used as a cure for insomnia. The sheer volume of documentation required to manage a system's development can easily intimidate even veteran project teams, often provoking "shortcuts" rationalized by the illusion that the methodology doesn't apply to the specific software project, or that the methodology itself is too cumbersome. But you don't have to let document flight decimate your development team—instead, try a solution that's as simple as a few scrawled lines on a page.

A Picture's Worth...

When quality teams first meet with project teams to identify a quality testing strategy, a mind map is an effective tool to help determine project scope. In lieu of a lengthy document, this concise graphical depiction of a project's testing scope can identify test case scenarios and distinguish different testing types early in the project life cycle.

Mind maps are merely network diagrams that utilize nodes, branches, colors and icons to convey an idea, offering a simple graphic view of a concept. We've all heard the adage that a picture's worth a thousand words—trite but true, because people often think and communicate in images. Written words build an image in the mind, and that image conveys an idea at second hand. But a graphic communicates that imagery directly, without the interpretation, parsing and extraneous elements inherent in any long text document. Think back to any process flow, organization chart or graphical timeline—they all convey a message, very directly, using pictures.

In testing, a mind map can easily depict a testing strategy by using its graphical format. For example, Figure 1 shows a testing mind map for an ATM system development project.

The mind map shown in Figure 1 grew out of multiple conversations among the project manager, business analysts and other team members, spurring review of project requirement documents, use cases and business process flows. The testing strategy's scope was quickly determined with a discussion of requirements for the system, and the group also identified both functional and nonfunctional tests.

Mind maps are methodology independent, benefiting both classic functional decomposition methodologies and iterative approaches that employ use cases. In both methodologies, the mind map identifies test cases, and team members can brainstorm alternative testing paths. The successful path tests are easiest to identify. Identifying the alternative testing cases will prevent programmers and system analysts from staying up all night chasing bugs after a system launch.

With a mind map, the QC analyst can more easily document testing strategy, and the project team can gain a graphical summary of the conversation. Despite structured agendas, human communication is often nonlinear, splitting into different tangents. The mind map is a useful method to capture those valuable detours that converge from the conversation's original direction.

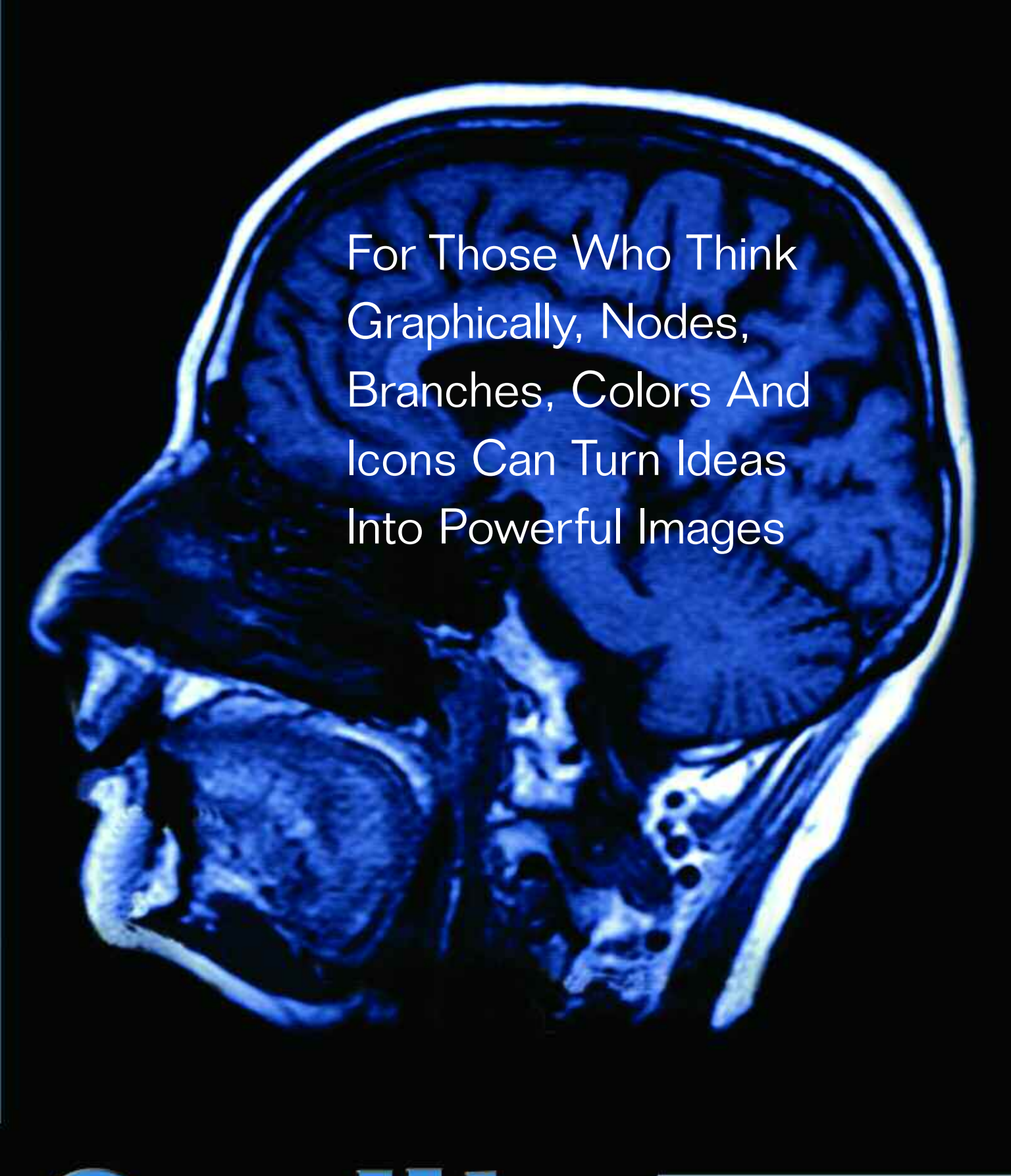
Once you start applying mind-mapping techniques, you'll find that they alleviate the ennui associated with writing notes, meeting minutes and project scope documentation. Despite methodologists' and project managers' affection for process and supporting paperwork, development teams universally abhor unnecessary documentation. Mind maps provide an innovative alternative to the "complete-this-document" approach.

Mind-Mapping Tools

A few sheets of paper and some colorful pens are all the equipment you need to develop a basic mind map, but you don't have to reinvent the wheel. Instead, why not explore some of the available software tools that help document, facilitate and communicate your mind map? Depending on your requirements, you can choose from among several commercial and free, open-source products. Here are some examples:

Mindjet MindManager (www.mindjet.com) is an easy-to-use tool that provides a repository of reusable mind-map branches called *map parts* that further help facilitate the mind-mapping process. An abundant number of icons is available to highlight test case opportunities or flag risks in the testing strategy. The product's integration with Microsoft Office is excellent and automatically generates PowerPoint presentations and Word documents

Photograph by Luis Carlos Torres



Quality

A PMI-certified Project Management Professional (PMP), **Andrew Makar** has worked in project management, design, implementation and maintenance of IT solutions since 1996. Reach him at andy@amakar.com.

"Best concentration of

Announcing The Third Annual

Software Test & Performance CONFERENCE

November 7-9, 2006

The Hyatt Regency

Cambridge, MA

The Software Test & Performance Conference provides technical education for test/QA managers, software developers, software development managers and senior testers.

The wall between test and development has fallen, and the Software Test & Performance Conference brings the whole application life cycle together!

BUT DON'T TAKE OUR WORD FOR IT...

"This is the best conference I have attended. The instructors were extremely knowledgeable and helped me look at testing in a new way."
—Ann Schwerin, QA Analyst
Sunrise Senior Living

For Sponsorship and Exhibitor Information Contact David Karp at 631-421-4158 x102 or dkarp@bzmedia.com

Software Test & Performance

Produced by

ECLIPSE review

SD Times — Software Development — The Industry Newspaper for Software Development Managers

BZ Media

TEST & QA Report

"If you get one impact idea from a conference it pays for itself. I got several at the

performance testing presentation/professionals I've seen." – Nathan White, Manager, Testing Services, AG Edwards

- **OPTIMIZE** Your Web Testing Strategies
- **LEARN** How to Apply Proven Software Test Methodologies
- **NETWORK** With Other Test/QA & Development Professionals
- **ACHIEVE** a Better Return on Investment From Your Test Teams
- **GET** the Highest Performance From Your Deployed Applications

"This is a conference to help both testers and developers as well as managers and leads. There is enough variety and content for everybody."
—*Michael Farrugia, Software Engineer*
Air Malta

"Great topics – well presented by reputable presenters. Having attended two years in a row, I have yet to be disappointed."
—*Ardan Sharp, QA Manager*
SunGard

"Excellent conference – provided a wide range of topics for a variety of experience levels. It provided tools and techniques that I could apply when I got back, as well as many additional sources of information."
—*Carol Rusch, Systems Analyst*
Associated Bank

"This conference is great for developers, their managers, as well as business-side people."
—*Steve Margenau, Systems Analyst*
Great Lakes Educational Loan Services

"I've received volumes of new information and ideas to share with my team."
—*Theresa Harmon, Business Applications Developer*
Pharmacare Specialty Pharmacy

"Reputable speakers and presenters. Great class topics."
—*Jung Manson, QA Manager*
Webloyalty.com

"Very informative and a good chance t[illegible]etwork with others."
—*Marty[illegible].Technic[illegible]ltant*
Autod[illegible]

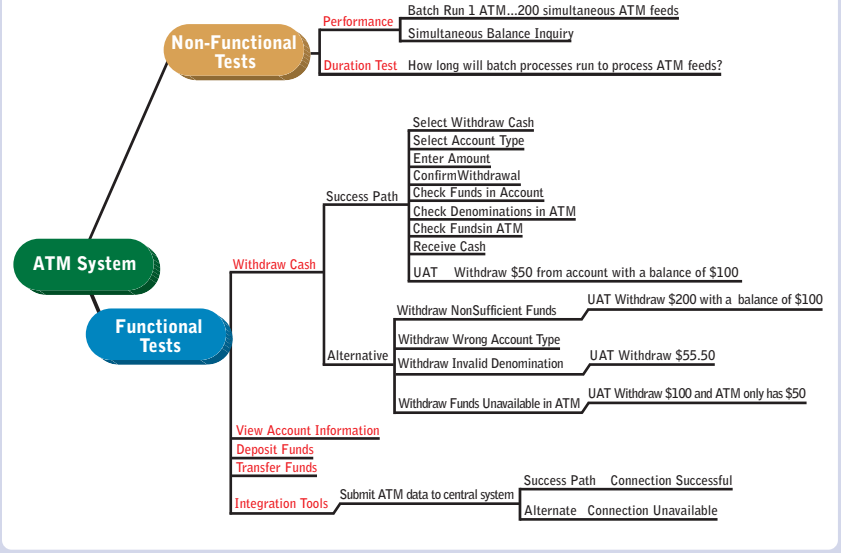
"A conference with something for everyone."
– Scott L. Boudreau, QA Manager, Tribune Interactive

Register by JULY 28 to take advantage of the eXtreme Early Bird Rates!

REGISTRATION IS NOW OPEN
www.stpcon.com

ST&P Conference." – Patrick Higgins, Senior Software Test Engineer, SSAI

FIG. 1: TEST-STRATEGY MIND MAP USING MINDMANAGER



describing the testing strategy. From a project management perspective, the ability to mind-map the test case strategy and export to a structured MS Project plan is a desirable feature. The company offers a professional license for $349 and a basic edition for $229. The professional version provides the entire mind-mapping functionality found in the basic version and provides integration with Microsoft Excel, Outlook, MS Project and Visio.

Visual Mind (www.visual-mind.com) is a commercial software package with a business and basic edition. The business edition provides additional features including Word and Excel document creation, file embedding, security features and multiple map tabs. The basic edition provides the core mind-mapping functionality; however, the business edition offers all the features that support publishing and exporting maps to different documents. A business license is $299, and the basic license is $99.

For Pocket PC enthusiasts, Pocket Mindmap (www.pocketmindmap.com) from JKRB Software is available for palm-sized computing. Pocket Mindmap supports multiple views, including mind map and outline formats. It also integrates with Pocket Outlook's task list and exports using an XML format. Other mind-mapping tools that support XML (such as MindManager) can import the map for future revision. At $42 per license, it may be a bit pricey for a palm-sized application, but Pocket PC fans will enjoy the mind-mapping mobility and leave their laptop or desktops at home.

Fans of open-source technologies will appreciate FreeMind (FreeMind.sourceforge.net) for its impressive features and zero cost. FreeMind's user-friendly interface allows you to quickly build mind maps while supporting multiple export features, including JPG, HTML, XML and the Open Office Writer format. Although the tool doesn't integrate with MS Office products, the integration points will converge as open-source and commercial software adopt open standards such as the Open Document format. FreeMind's zero cost and easy-to-use features make it a great productivity tool. Figure 2 depicts a quality mind map created with FreeMind.

CmapTools is another free concept-mapping tool, available at cmap.ihmc.us. Sponsored by the Institute for Human and Machine Cognition, CmapTools enables users to construct, navigate, share and collaborate on mind maps. The site's unique structure shows visitors how mind maps convey knowledge. The collaboration features are appealing if QC organizations want to collaborate and share ideas on quality test cases. The tool supports a number of export formats, including text, image, XML and HTML.

After evaluating several tools, I found Mindjet's MindManager and FreeMind to be my personal favorites. MindManager's integration feature with Microsoft Office ranks it high on my list of office productivity tools. Users will appreciate exporting a mind map into an MS Word document, MS Project plan, or MS PowerPoint presentation. Quality managers can export potential test-case scenarios into a presentation with a few mouse clicks.

FreeMind's sheer simplicity, low learning curve and zero-cost status make it an attractive tool for organizations using open-source tools. FreeMind is not as function-rich as MindManager, but for basic tasks, it's an excellent choice. CmapTools is a close second, although FreeMind's learning curve is far shorter.

All of the commercial sites listed offer free evaluation copies, so gather up a few packages and play with them for awhile to choose the one that works best for you and your organization's budget.

Recommended Reading

You don't need a software package to jump into mind mapping: For more

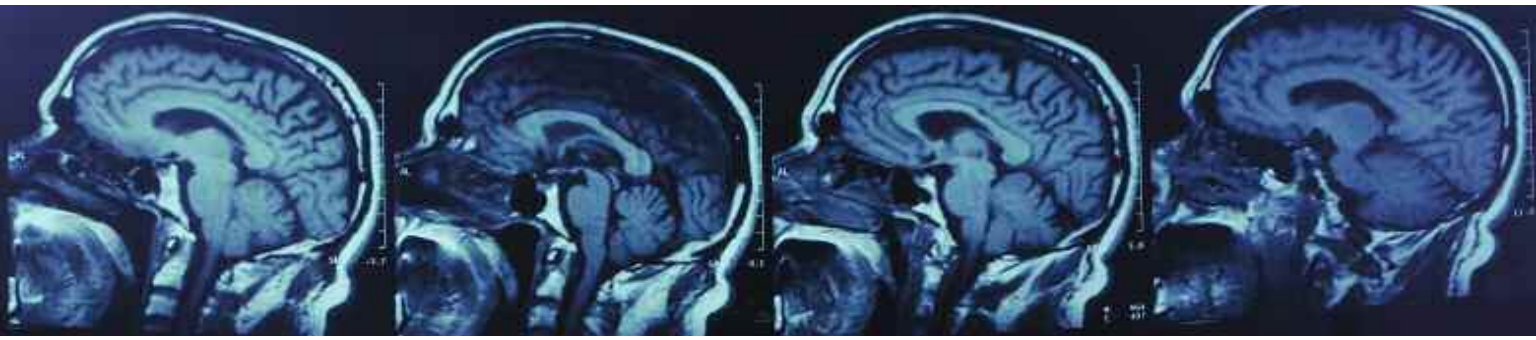
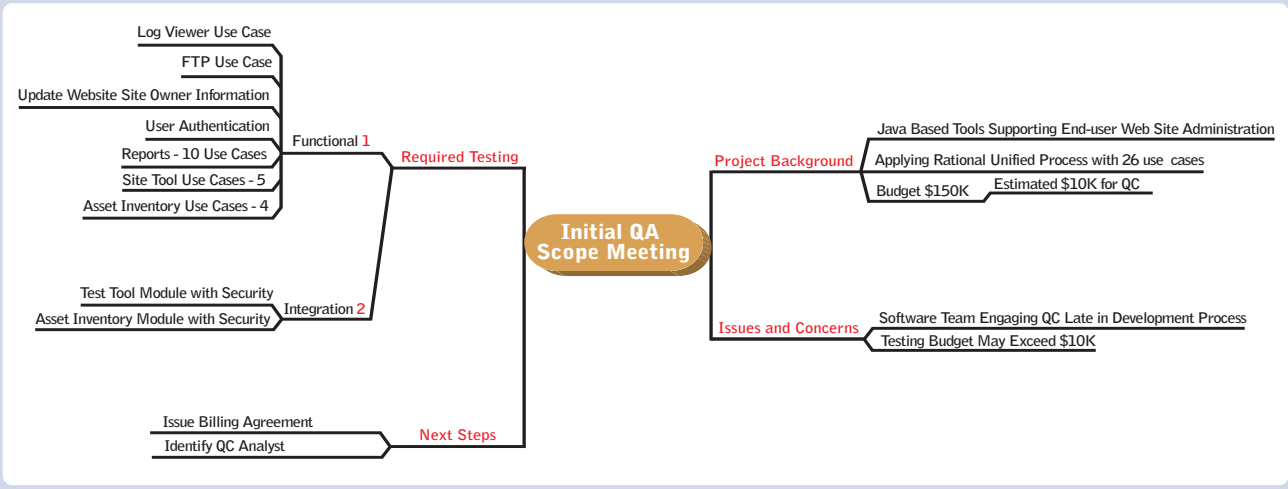


FIG. 2: QC SCOPE-MEETING MIND MAP USING FREEMIND



information, you can turn to several books that will help jump-start your creative thinking and brainstorming. According to Wikipedia, author Tony Buzan is a mind-mapping pioneer aiming to "improve learning and clearer thinking that will enhance human performance." His books include:

- "The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential," Plume reprint edition, 1996
- "Mind Maps at Work: How to Be the Best at Your Job and Still Have Time to Play," Plume, 2005
- "How to Mind Map: Make the Most of Your Mind and Learn to Create, Organize and Plan," Thorsons, 2003

Train Your Brain on the Net

You're only a few Google keystrokes away from finding a plethora of useful mind-mapping sites. Here are a few recommended links exploring mind mapping:

InnovationTools' Mind Mapping Resource Center at www.innovationtools.com/resources/mindmapping.asp has a number of useful links using multiple software tools.

Blog enthusiasts can learn the latest news and trends in mind mapping at mindmapping.typepad.com. The blog provides multiple tips and techniques, recent news, books and input from more than 20 software vendors providing mind-mapping solutions.

Proving that the iPod and podcasting have dominated the MP3 market, blog.mindmap-software.com provides more than 25 mind-mapping podcasts for your listening pleasure. Instead of rocking to the latest Dixie Chicks' offering or working out to Taylor Hicks' upcoming "American Idol" release, you can train your brain by listening to the latest in mind-mapping podcasts.

The application of mind maps isn't limited to QC testing—it also includes strategies for project management, information management and strategic planning. Visit www.mindmappingstrategies.com for more information and additional links.

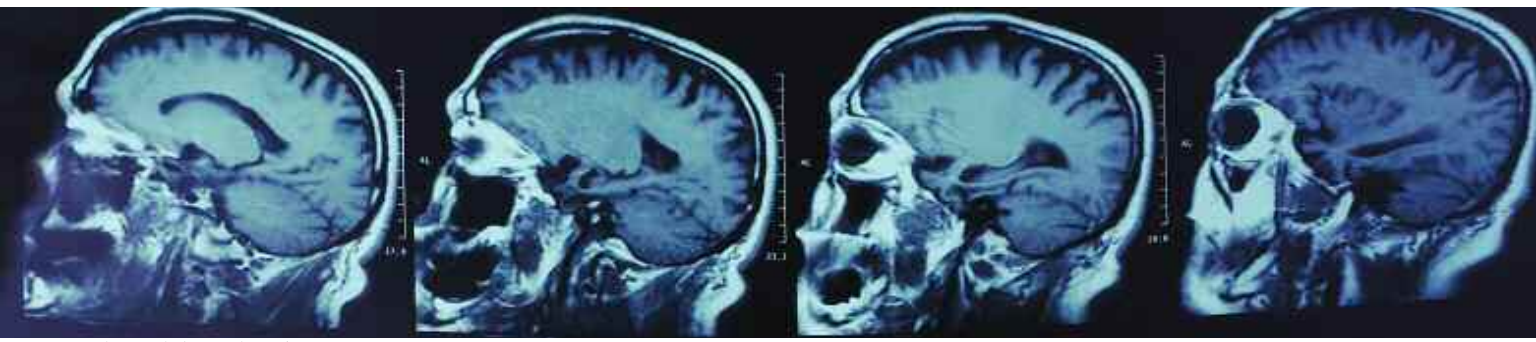
Quality Management and Beyond

Once you start applying mind maps to your quality management work, you'll quickly find them relevant in other avenues of your professional and personal life. Mind maps are useful in project management to identify project tasks, assign people, and capture issues and risks during project execution. An excellent alternative to drab meeting minutes, mind maps can highlight creative ideas and prioritize action items.

Instead of delivering a speech using note cards, use a mind map to identify key points. Replace a weekend task list with a mind map of weekend activities. Refresh a grocery list with a mind map of products and their related aisles.

For the QC analyst, mind maps are effective tools to define a project's testing scope, identify potential testing issues and risks, determine tasks and document decisions.

Scope management is always a challenge in IT delivery because every new feature requires multiple test cases. If QC managers, business analysts and project managers apply mind maps to further define and communicate scope, projects are more likely to succeed. Start with a few mind maps, and you'll soon be mind-mapping your way to quality.



Photographs by Emrah Turudu

Early and

You Can Seize Control Of Your Testing Process With Better Use Of Resources

By Robin Goldsmith

Testing is our primary means of controlling system and software risks. Every testing authority explicitly states that testing should be in proportion to the degree of risk involved: The greater the risk, the more testing is needed. For example, the IEEE SWEBOK software engineering body-of-knowledge section on testing states, "Testing must be driven based on risk; i.e., testing can also be seen as a risk management strategy." (Version 1.0, May 2001, page 75)

Conducted correctly, risk-based testing enables better use of limited time and resources to ensure that the most important work is completed and that any tests that are skipped or shortchanged are of lesser import.

Planning Your Process

To determine which test cases are the most deserving of your available time and resources, your testing process should be planned and designed, largely through identifying, prioritizing and addressing potential risks.

To be confident in your testing process, you must think systematically about what needs to be demonstrated, rather than depend on the busywork paper-pushing that many seem to confuse with planning. Value is maximized by economically and appropriately documenting risks so they can be remembered, shared, refined and reused. Write no more than is useful—and no less.

Risk exposure represents the combination of impact (damage, if the risk does occur) multiplied by likelihood (that the risk will indeed occur). Much of risk literature involves variations on a useful but mechanical method for quantifying risk: assigning respective values to impact and likelihood, such as 1 = Low, 2 = Medium, and 3 = High, and then multiplying to produce a risk exposure score of 1 (very low) through 9 (very high). Figure 1 (page 26) shows the respective exposures graphically: Low = 1 box, Medium = 4 boxes and High = 9 boxes. As with most images, these graphic depictions can enhance understandability.

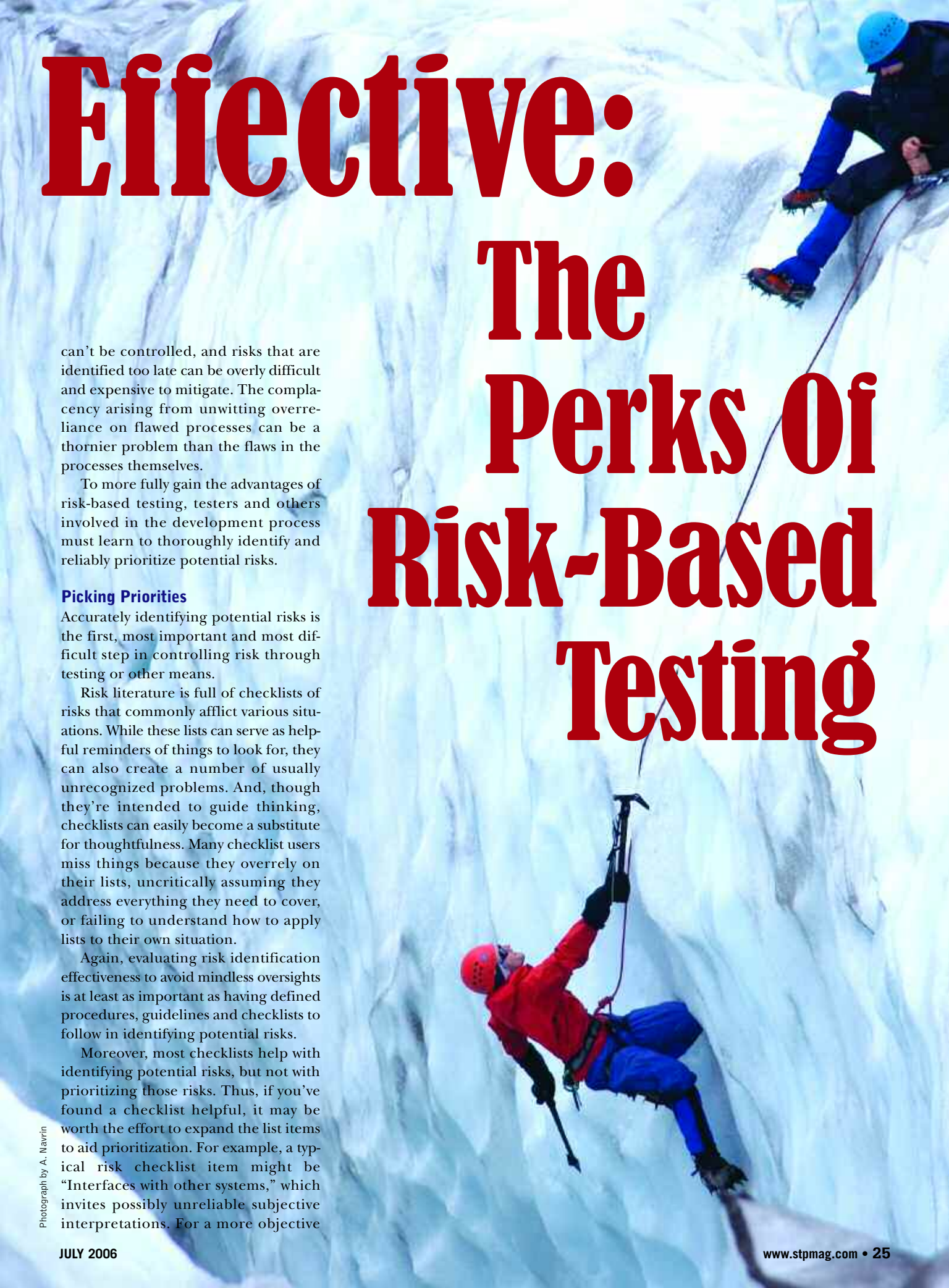
However, pictures can be misleading: Some testers may adjust the scale as a form of gamesmanship. For instance, values of 1 = Low, 3 = Medium and 5 = High yield risk exposure scores of 1 (very low) through 25 (very high). As Figure 2 (page 27) suggests, an exposure of 25 boxes can seem far greater than an exposure of nine boxes, although a maximum 25 score on a 1–25 scale actually represents a risk identical to a maximum 9 score on a 1–9 scale.

Virtually everyone agrees on the concept of basing testing on risk impact and likelihood. However, testers continually encounter difficulties in effectively applying risk-based testing in practice. In explanation, many testing authorities imply that the way to identify and prioritize risks is obvious, and that too often, overly simplistic examples make a task seem deceptively easy—until you try to put the techniques in practice with a real system.

An even more significant hidden risk to the development and testing process occurs when testers are unaware of the possibility that they may not be adequately identifying and prioritizing risk-based tests. Risks that aren't identified

Robin F. Goldsmith, JD, is President of Needham, Mass.–based consultancy Go Pro Management, Inc. A prominent speaker and author of "Discovering REAL Business Requirements for Software Project Success" (Artech House, 2004), he advises and trains systems and business professionals. He can be reached at robin@gopromanagement.com.

Effective: The Perks Of Risk-Based Testing



can't be controlled, and risks that are identified too late can be overly difficult and expensive to mitigate. The complacency arising from unwitting overreliance on flawed processes can be a thornier problem than the flaws in the processes themselves.

To more fully gain the advantages of risk-based testing, testers and others involved in the development process must learn to thoroughly identify and reliably prioritize potential risks.

Picking Priorities

Accurately identifying potential risks is the first, most important and most difficult step in controlling risk through testing or other means.

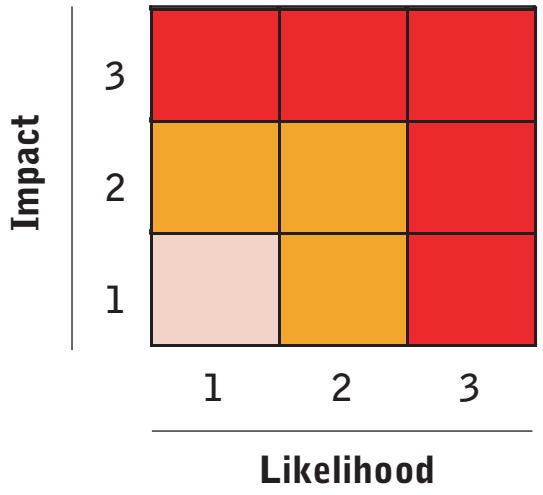
Risk literature is full of checklists of risks that commonly afflict various situations. While these lists can serve as helpful reminders of things to look for, they can also create a number of usually unrecognized problems. And, though they're intended to guide thinking, checklists can easily become a substitute for thoughtfulness. Many checklist users miss things because they overrely on their lists, uncritically assuming they address everything they need to cover, or failing to understand how to apply lists to their own situation.

Again, evaluating risk identification effectiveness to avoid mindless oversights is at least as important as having defined procedures, guidelines and checklists to follow in identifying potential risks.

Moreover, most checklists help with identifying potential risks, but not with prioritizing those risks. Thus, if you've found a checklist helpful, it may be worth the effort to expand the list items to aid prioritization. For example, a typical risk checklist item might be "Interfaces with other systems," which invites possibly unreliable subjective interpretations. For a more objective

Photograph by A. Navrin

FIG. 1: RISK PRIORITIZATION



analysis of interface risks, the item could be rewritten as:

Low risk = 0 interfaces with other systems
Medium risk = 1–2 interfaces with other systems
High risk = 3 or more interfaces with other systems

Checklists have a tendency to inadvertently intermix perspectives, which can seriously interfere with the prioritization process. Apples-versus-oranges confusion can be reduced by consciously conceptualizing risk identification perspectives and systematically approaching each perspective individually. Many published risk checklists reflect a dominant perspective that could be refined to facilitate apples-versus-apples comparisons.

The bulk of the literature on risk comes from a management perspective. People with engineering and operations orientations, including testers and non-engineering folks such as many business users, tend to concentrate on risks from a product/technical perspective. In part because generalized lists are unlikely to address an individual situation's specifics, business effects comprise the perspective that is most commonly neglected—and which, ironically, carries the most significant risks of all.

Business Effect Risks

The common failure to sufficiently recognize, understand and appreciate the importance of business effect risks virtually guarantees that all other risk-control efforts (such as testing) will be less effective.

Therefore, the first and most important test-planning and design risk-identification step is to determine the effects on your business if the system/software doesn't work as needed. Note that the issue is *what* the business requires, not product/system/software requirements that dictate *how* a presumed product or system solution is intended to work. Business effect risks answer the question, "So what if such and such happens—or doesn't?"

Not only are we unaccustomed to thinking in business terms, even when we do, we may have difficulty thinking about effects. Thus, despite conscious direction to identify business *effect* risks, people tend to instead identify actions and events that are presumably *causes* of some damaging effects, though those effects are generally not articulated.

Intermingling risk causes and effects, which is a common checklist approach, also inhibits meaningful prioritization, because comparisons must be made among comparables. Since risk is an effect, it's essential that impact/likelihood prioritization be focused entirely on potential risk effects. On the other hand, mitigation approaches are based on those effects' causes, which in turn must be clearly understood.

Business effects are the most important of all risks because they provide a context for analyzing all risks. Without such a context, it's easy to miss the most important management and technical risks, and divert your attention to more visible but perhaps less important areas.

Management Risks

Much of the risk literature deals with management risks, such as lack of resources and time, long duration, large projects and faulty development processes, including poor estimation, inadequate testing and requirements changes. Since 9/11, articles on risk have tended to deal with security breaches, although in the aftermath of Hurricane Katrina, the emphasis has shifted somewhat to the risks of natural disasters.

The most common published checklists come from a management risk perspective, and almost all such lists unwittingly intermingle causes and effects. Attempting to use these checklists for risk-based testing poses several additional pitfalls.

Checklists can inadvertently mix perspectives, which can interfere with the prioritization process.

First, except for a limited set of specifics such as security and disaster recovery, testing may not be a relevant response to mitigate many management risks. For instance, an understaffed project with an overly aggressive deadline will undoubtedly encounter quality issues, partly because testing has probably already been squeezed. More testing in general will help, but such general management risks don't help you determine *which* testing to increase.

Second, risk involves statistical probability. However, for most organizations, many of the typical management-risk checklist's items aren't risks, but certainties. Practically every project is underestimated, usually by a large percentage that grows as requirements appear to

change, always in ways that necessitate additional time and resources. Incorrect estimates result in allocating insufficient or inappropriate time and resources, which regularly leads to deliveries that are late (or nonexistent), over budget, and wrong.

Inadequate testing is both a symptom and a cause of the predictable problems of typical development practices. Squeezing testing is a common management response to underestimated projects that are running late, but being late also stresses development and produces still more errors, which the squeezed testing is in turn incapable of catching. By definition, such faulty processes will certainly continue to create the conditions that produce problems. Traditional risk-based testing is highly unlikely to be the means for ensuring the time and resources necessary to break this cycle.

Third, testers tend to approach matters from the bottom up, often mainly in terms of executable test cases, and checklists seldom recognize the interrelationships of various management risks. This often quashes awareness about the ways that testing could in fact help mitigate the conditions cited on management risk checklists.

For example, the main reason for inadequate budgets, resources and schedules is inadequate task definition and estimation, which stems mainly from inadequate understanding of the work to be done, which is generally caused by inadequate product/system design, which is mainly due to inadequate definition of the real business requirements.

Failure to adequately discover the real business requirements is often caused by a lack of awareness of their nature and of the vital need to identify them in detail. Instead, conventional wisdom concentrates on the system/software/functional requirements, which actually comprise a high-level design of the product/system that is *presumed* to be what is needed to satisfy the *presumed* but usually undefined real business requirements. Table 1 summarizes these distinctions. Focusing solely on any downstream issue such as high-level design addresses symptoms rather than causes.

By recognizing how these management risks are interrelated and largely stem from inadequate requirements, appropriate testing can be a powerful aid to mitigation through detecting requirements and design issues early, when they're easiest and cheapest to fix.

Reviews

Reviews are the method used for "testing" or evaluating the adequacy of earlier development deliverables, such as requirements and designs. Many organizations consider reviews to be quality assurance rather than testing. Regardless, while certainly helpful, conventional requirements and design reviews tend to be far weaker than is usually recognized.

Some organizations don't review their requirements and designs at all; those that do ordinarily use only one or two weaker-than-realized techniques. Most conventional reviews miss many of the most important issues because they erroneously focus on product/system/software requirements' high-level design, rather than on real business requirements that provide value. Also, conventional reviews often concentrate mainly on matters of form, such as clarity and testability, rather than substance. And too often, the rest of the organization disregards such format-oriented review results as trivial nitpicking.

In contrast, Proactive Testing (see "Mindset and Methodology" sidebar, page 30) can break this vicious cycle, with more than 21 ways to evaluate the adequacy of business requirements, and more than 15 ways to evaluate designs. Many of these are more powerful, content-oriented techniques that also spot the overlooked and incorrect requirements and designs that format-oriented reviews tend to miss.

Overlooked and incorrect business requirements often relate to significant business effect risks. Business and management who appreciate the importance of these issues can look to Proactive Testing to spot red flags before they become predictable problems.

Moving Beyond Reactive Testing

Testers tend to come at risks from the perspective of the product or system being developed. Conventional risk-

FIG. 2: RISK PRIORITIZATION GAMESMANSHIP

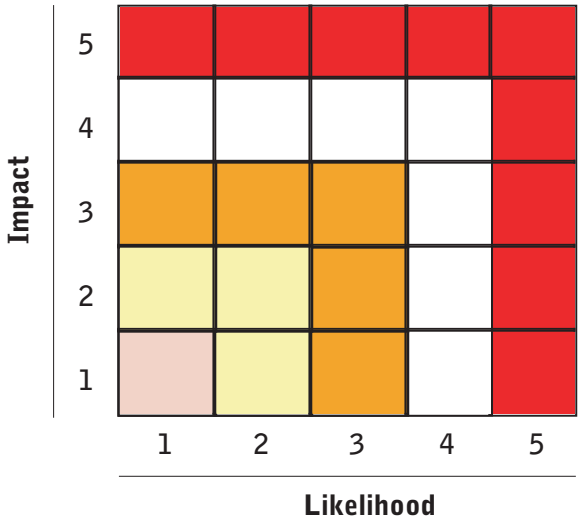


TABLE 1: TWO TYPES OF REQUIREMENTS

Business/User Requirements	Product/System/Software Requirements
Business/user perspective and language, conceptual.	*Human-defined* product or system perspective.
Exists within the business environment and thus must be discovered.	One of the possible ways (high-level design) to accomplish presumed business requirements.
When delivered, accomplished or satisfied, provides value by serving business objectives, typically solving expected problems, meeting challenges or taking opportunities.	Often phrased in terms of the external functions that the product/system is expected to perform; thus also called *functional specifications/requirements*.

based testing, as espoused by essentially every testing authority, analyzes a system's design to identify risks related to the system's functionality/features (which are sometimes called *requirements risks*) and/or the system's technical structure (its components and how they're put together), and then tests the higher risks more frequently.

Such risk identification may be done individually or in a group. Ordinarily, the system's functionality is defined based on user interface design, where each menu or GUI button choice is deemed a separate feature to analyze for risk. To identify structural risks, organizations often have a wider variety of technical design artifacts. For example, architecture and network diagrams, database structures and object models each may be reviewed to identify different types of component risks.

While sometimes guided by checklists, product/technical risk reviewers are often simply directed to apply their experienced judgment to identify those features and components that will have a significant impact if they go wrong, as well as those that seem likely to go wrong.

For each test-design specification to be tested, a set of test-cases is identified that will prove that the feature, function or capability really works.

Product/technical risk checklists generally prompt attention to things that are new or changed, complex or large, or used frequently. They also focus on interfaces, dependencies and historical problems. While definitely good advice, such checklists are often difficult to apply to particular systems, especially with regard to functionality.

This conventional risk-identification approach is necessary and important, but is generally less effective than realized because it's reactive. One reason for this reactivity is that, much to their frustration, testers often don't come on board until the end of the development process, after problems have already been incorporated in the code. And, in a side effect of such reactive late involvement, risk identification can become overwhelming, with masses of details dumped on the testers all at once. Without effective ways to get a handle on the whole thing, it's easy to get mired in detail and lose sight of bigger issues.

In addition, the greatest weakness of most conventional testing is that risk identification merely reacts to and therefore tends to be limited to whatever's in the design or code. Being guided by the design—or, even worse, the code— can lead risk identification away from recognizing errors and omissions.

The Proactive Plus

In addition to more effective reviews of requirements and design, Proactive Testing includes but goes beyond conventional testing's reactive product risk identification. Taking business and management as well as product perspectives, Proactive Testing uses specific techniques that help identify many important risks that reactive approaches miss. Moreover, Proactive Testing addresses risks earlier and at multiple points in the development process, scaled to need.

Consequently, whereas conventional reactive testing enables more frequent testing of higher risks, Proactive Testing enables testing higher risks not just more often, but earlier, when problems are easier to fix.

Proactive Testing can employ the approach suggested by IEEE Std. 829-1998 to organize the test planning and design process. When applying this proven project-management practice, testing is systematically broken down into smaller, more manageable pieces. At each point in the process, risks are identified and prioritized to guide a

successively narrower focus on the most important areas to test.

This structure provides a way of thinking, not a mandate for generating paperwork, as some people (mis)interpret the Standard. However, it's important to write information down so it can be remembered, refined and reused. In Proactive Testing, you write as much as is helpful—no more, but no less.

The earliest and most important Proactive Testing risk analysis occurs as part of master test planning, which is performed most effectively as soon as you have a high-level system design.

The master test plan is the project plan for the testing project, which is a subproject within the overall development project. Each key risk to the system being developed is addressed in a detailed test plan—for each unit, integration, system and special (such as load or security) test. Detailed test plans dealing with the highest risks should be tested more often and earlier in the life cycle.

Powerful Proactive Testing techniques identify the biggest risks, including up to 75 percent of the showstoppers that conventional reactive testing ordinarily overlooks because typical development processes have failed to address these critical risks in the system design. Moreover, Proactive Testing identifies the risks early enough to let testing drive development. That is, while it's important to merely catch showstoppers prior to production, Proactive Testing can increase value dramatically by catching the showstoppers in time to prevent writing affected related code that otherwise must be thrown out and rewritten.

The value of spotting these ordinarily overlooked risks is greatest in conjunction with high-level design, but remains significant throughout development. Payback from preventing a showstopper risk is always valuable, even if it's not found until the day before the show would have been stopped.

While checklists can help spot generic types of risks that afflict many systems, Proactive Testing also emphasizes understanding the specifics in order to reveal the system's unique big risks, which are perhaps the most commonly overlooked. The written risk findings can become a checklist for future risk analyses, but beware that overreliance on checklists can prevent the meaningful, content-based risk identification that Proactive Testing uses to spot so many otherwise overlooked risks.

It's especially valuable to involve a broad range of participants in identifying these big risks, since Proactive Testing facilitation prompts each different view to identify risks that other participants would probably miss. Experiencing the discovery of big risks wins instant advocates for early, Proactive Testing involvement. In addition, a group representing a broad mix of involved stakeholders provides the most effective risk prioritization.

Because Proactive Testing identifies big risks early, test plans dealing with the higher risks can be tested not only more often, but earlier in the life cycle.

Scaling and Carrying Through

Each of these big-risk areas can be addressed in a detailed test plan, which should be just elaborate enough to be helpful; no more, but no less. The higher the risk involved, the more analysis will be needed. Lower-risk areas can be given respectively less attention, even if only to the point of documenting the conscious decision not to analyze them further.

Because Proactive Testing identifies these big risks early, detailed test plans dealing with higher risks can be tested not only more often, but earlier in the life cycle.

To do this, development must structure and schedule its work to create the higher-risk components earlier so they can be tested earlier. One powerful way to do this is to define the sequence of builds based at least in part on Proactive Testing risk prioritization.

MINDSET AND METHODOLOGY

Proactive Testing is as much a mindset as a methodology for risk-based software test planning, design and management. Positioning testing within an overall quality and business value context, Proactive Testing emphasizes WIIFMs (What's In It For Me) that win over users, managers and developers. Rather than the traditional adversarial perception of testing as an obstacle to delivery, Proactive Testing can save you aggravation while helping you deliver better systems cheaper and quicker.

The Proactive Testing life cycle embraces true agility to minimize wasted effort by enabling the development process to build more right in the first place. At the same time, by intertwining more thorough and effective tests of each development deliverable at multiple key points, it can economically detect more errors.

Proactive Testing uses higher-payback test-planning and design methods that augment conventional testing techniques to find numerous test conditions commonly overlooked by traditional testing methods, allowing more effective scaled risk prioritization and repeated refocusing on the most important tests. Moreover, anticipation can promote reuse and let testing drive development to prevent problems or catch them earlier, when they're easier to fix.

By maintaining a real process perspective, Proactive offers a fresh take on conventional User Acceptance Testing (UAT). Rather than a subset of technical testing (unit, integration, system/special), UAT should be kept independent. With truly proactive, user-centered planning/design of UAT up-front for execution at the end, Proactive Testing can help improve requirements and designs while providing a more thorough and effective double-check of the developed system.

Finally, Proactive Testing accepts responsibility for results, which includes meaningful measurement and active management.

Each detailed test plan identifies the set of test design specifications that, taken together, give assurance that the subject of the detailed test plan works. Each test-design specification describes the set of test cases (inputs/conditions and expected results) that must be conducted to ensure confidence that a feature, function or capability works.

Just as Proactive Testing methods identify numerous otherwise-overlooked big-risk areas, additional special techniques successively identify many risks in ordinarily overlooked features, functions and capabilities. Conventional methods that fail to identify these risks can't include them in risk prioritization or mitigation.

The fully identified set of test design specifications is then prioritized according to the risks they address. Those dealing with the highest risks are tested earlier and more often. In addition, by structuring test-design specifications to be reusable, Proactive Testing offers an additional way to increase the amount of testing in the time available.

For each test-design specification to be tested, a set of test cases is identified that will prove that the feature, function or capability works. The test cases are ranked based on the risks they address, with the highest-risk test cases executed earlier and more often.

Prioritization Alternatives

It's common to use the well-known impact-multiplied-by-likelihood high/medium/low ratings risk-prioritization approach described above. However, this method frequently takes too much work and is too difficult to achieve consensus on, especially with a broadly representative group of stakeholders.

A further concern with the technique is that rating each risk individually (as does the commonly used mandatory/desirable/ideal differentiation technique) can actually prevent effective prioritization because each risk viewed in isolation tends to be rated high.

By definition, prioritization demands distinctions among risk choices, so it's advisable to use a risk prioritization technique that produces a ranking. As a technique, ranking raises two further issues.

First, for meaningful ranking, your selections must be limited to a manageable number, typically no more than 15 and preferably closer to seven (a range that numerous scientific studies have found to be the parameters of the typical human attention span).

Second, since rank alone doesn't convey relative differences in importance, it's more valuable to use a risk analysis technique that enables ranking based on quantified importance. Such methods can be gross or precise; formal or informal.

The simplest technique is to gain group consensus on a relatively small subset of risks that are very important relative to the fuller set of all identified risks. The subset of highest risks can then be ranked, or ranking can be skipped if it's certain that all the subset members will be addressed.

You can also try a more formal technique: the *100-point must system*. Each participant is given 100 points that he must allocate among the risk choices in proportion to importance. All 100 points must be allocated, and at least one point must be allocated to each risk choice. Wide discrepancies are then discussed and adjusted as appropriate, followed by arithmetic averaging of respective risks.

Effective risk-based testing doesn't end with risk identification and prioritization.

Managing the Process

Effective risk-based testing doesn't end with risk identification and prioritization. Tests of the highest risks must be planned, designed and executed. As time and resources allow, lower risks must also be tested.

Most importantly, testing effectiveness must be monitored and measured, issues must be detected in a timely manner, and appropriate actions taken.

Ultimately, the time and cost of testing must be weighed against its benefits. Did the testing catch problems that would have made a significant impact? Would the problems have been found without risk identification and analysis? Would the costs of those problems' occurrence exceed the costs of detecting them? What kinds of damage occurred in spite of risk-based testing? And what additional/different risk analysis and/or testing would have prevented the damage?

Using History to Help Topple Problems

By Prakash Sodhani

Despite the abundance of automation tools on the market, many organizations rely completely on manual testing techniques, using Microsoft Word documents and Excel spreadsheets to create test cases and document test-case execution results. From these documents, various testing metrics are derived and decisions made. And, because of the ubiquity of manual testing, a solid understanding of the entire testing life cycle is essential.

In each phase of that life cycle, metrics must be captured as needed. In the execution phase, actual results are compared against the expected outcome. Test execution varies wildly, ranging from a handful of test cases for a small project to thousands for a much larger work. But whether humble or grandiose, like any life cycle, the test case process encompasses a series of identifiable, sequential steps.

The *test-case execution states* metric describes a test case's various states during the execution phase In a test case's life cycle, the most common execution states are Passed, Failed, Blocked and Deferred. Test-case execution states can be a rich source of vital information, helping you to make decisions that ultimately benefit your project as a whole. In this article, I'll describe a typical test-case life cycle and explain its significance with an example.

Making Metrics Matter

Metrics serve as a project road map. Because they detail information about project status at each step, metrics help to identify issues as they occur—while you're able to respond to them. Without metrics, a project is like a traveler who has no idea how to reach his destination—and may never arrive there.

Metrics vary with the organization and depend on various factors, such as the project's overall nature, the individual phase involved and the client requirements. They should be captured at each phase of the project, in the requirements, design and testing phases.

Some of the more commonly used testing metrics include the number of test cases written, the number executed and the number passed. With so many testing metrics available, it's easy to neglect some, despite their importance. But with close attention, these measurements—including test-case execution states—can play a vital role in making key project decisions.

The Test-Case Life Cycle

During execution, a test case can exist in different states, depending on various factors. A test-case life cycle consists of various test case states during execution. Four of the most common states are these:

- ***Blocked:*** (Denoted as *B*) In this state, test case execution can't proceed because of dependency on other factors, such as absence of test data.
- ***Deferred:*** (Denoted as *D*) Test case execution is postponed; for example, desired application function-

Prakash Sodhani is a longtime QA specialist who has worked in a number of IT organizations in different capacities as a quality professional. You can reach him at Prakash.Sodhani@gmail.com.

Photograph by Dwight Lyman

ality isn't implemented on time by the development team.

- ***Failed:*** (Denoted as *F*) In this state, test-case execution results don't match the expected outcome.
- ***Passed:*** (Denoted as *P*) In this state, the actual result of a test case execution matches the expected result.

During its execution, a test case can typically progress through *F*, *B* and *P* states. For example, a test case can change state as follows:

F→ *F*→ *B*→ *F*→ *F*→ *P*.

The Deferred state implies that the case is not of immediate consideration, and so usually occurs independently.

A test life cycle may involve the execution of a number of test cases, each of which undergoes varied execution states. Some reach the Passed status in their first execution, while others may go through any number of Failed, Deferred and Blocked states before they attain that status.

The obvious question arises: What is the importance of the intermediate states of a test case execution? Shouldn't we care only that the test case passes?

What is the importance of a test case execution's intermediate states?

Don't Repeat History

As philosopher George Santayana told us, those who don't remember the past are condemned to repeat it—and you certainly don't want to repeat the testing process any more than you have to. To this end, by designing your metrics to delineate a test case's history of blocks, fails, passes and other states, your results will be far more useful in determining exactly what went wrong where. Test cases are executed to achieve a short-term objective—uncovering errors—but they should also serve the long view. A test case that includes execution state history can be even more valuable on large projects, where time is a luxury and you can test only a sampling of representative functionalities of various modules.

Consider the following application functionality as it's being put through its paces by two different testers.

Specification: "The Web page should take less than five seconds to appear."

Tester1, concerned only with the final result, reports that the test case has reached the Passed status.

Tester2, who maintains a log of the test case's varying states throughout the test-execution phase, reports the following:

"Test case execution went through following states:

B→ *F*→ *F*→ *F*→ *P*. "

After an analysis based on those states, Tester2 reported:

"Initial execution of the test case was not possible because of test data and environment issues. Hence, the execution stayed in the Blocked state for a week. Once the test data and environment issues were resolved, I noticed that the average response time for the Web page to load was 10 seconds (twice the expected time frame), so the status then indicated Failed. In the next two executions, response time was close to 10 seconds, eliciting another Failed status. In the fifth execution, the load time for the Web page was five seconds, as expected, so the test case state was changed to Passed."

As Tester2's attention to history reveals, an ongoing log of test-case execution states can provide far more detailed information than a mere final result.

Testing an E-Commerce Application

Kate, a test engineer in a company *X*, is working on an e-commerce project. To test a feature of the application, she's written three test cases: T1, T2 and T3. Let's explore two alternative scenarios for the

TABLE 1: 1ST EXECUTION

Test Case	Status	Comments
T1	*F*	Please refer defect id: 320
T2	*B*	Need additional test data
T3	*P*	

TABLE 2: 2ND EXECUTION

Test Case	Status	Comments
T1	*P*	Please refer defect id: 320
T2	*B*	Need additional sample data
T3	*P*	

TABLE 3: FINAL EXECUTION

Test Case	Status	Comments
T1	*P*	
T2	*P*	
T3	*P*	

test-execution life cycle.

In our first scenario, Kate has completed the initial round of execution testing, as shown in Table 1.

The next day, Kate learns that defect id 320 has been fixed and is ready for testing. Although the data needed to test case T2 is still not available, she executes the test cases. Table 2 lists test case status after the second execution.

Finally, Kate is able to create the required test data and perform the final test execution. After all three test cases reach the Passed status, testing is supposed to be complete, and she updates the final grid with the test cases' status, as shown in Table 3.

Kate's metrics raise a few questions:

- What can be inferred from Table 3? Does it imply that all test cases were executed without any failure?
- Do we have any information about test case states throughout the execution phase? Would such information be useful?
- How can the data in Table 3 help Kate make informed testing decisions?

In our second scenario, Kate has just completed the first execution, as shown in Table 4. But this time, she adds an important detail:

In Table 4, I1, I2 and I3 refer to three test-execution iterations detailing the test case history of varying defect states as testing moves through three trials.

The next day, Kate learns that defect id 320 has been fixed and is ready for testing, although sample data needed to test case T2 remains unavailable. In a second iteration, she executes the test cases and updates test case T1's status to Passed, as shown in Table 5.

Finally, Kate is able to create the required test data and performs a third iteration. Table 6 depicts test case status after this iteration is complete.

After all three test cases reach the Passed status, testing is supposed to be complete. Table 6 reveals the final test-case status:

Kate's new scenario offers a more detailed view of the test-execution process:

- It reveals that T1 failed during the first iteration, and that some test data was needed for successful execution of T2. T3 was executed without any problems, while T1 and T2 required more iterations to reach the "Passed" status.
- It depicts the changing states of test cases during the ongoing execution cycle, revealing the associated issues. For example: T1's Failed status might imply that this test case had some inherent problems, while T2's Blocked state suggests that outside help for test data creation might effect successful execution.
- In the face of looming deadlines or a large project, when you must select only a representative sampling for regression testing, test-case execution history can be a vital tool in determining which test cases to execute.

Overheard at the Water Cooler

Let's eavesdrop on a typical conversation between two testers, Peter and Juanita:

"Our project is at a critical stage," Peter says. "It's scheduled to go into production in a couple of months."

"I know," Juanita replies. "We need to do a rigorous regression testing. And since we have hundreds of test cases, we've got to find a way to select a representative sample."

"Regression testing is a euphemism for 'ad hoc testing,'" Peter adds ruefully. "I guess we'll just pick test cases at random and get going on this."

"You think so?" Juanita asks, raising an eyebrow. "I'm not so sure about that.

TABLE 4: 1ST ITERATION

Test Case	Status	Comments
	I1	
T1	F	Please refer defect id: 320
T2	B	Need additional test data
T3	P	

TABLE 5: 2ND ITERATION

Test Case	Status		Comments
	I1	I2	
T1	F	P	Please refer defect id: 320
T2	B	P(??)	Need additional test data
T3	P		

TABLE 6: FINAL ITERATION

Test Case	Status			Comments
	I1	I2	I3	
T1	F	P		Please refer defect id: 320
T2	B		P	Test data created
T3	P			

Why not figure out a better way to select a sampling of the functionalities to be tested? Also, we should take into consideration those test cases that had issues and problems when they were executed during the regular test cycle."

Juanita is right—Peter's random approach will produce random results. Instead, a thorough history of test-case execution states can provide a vital tool for selecting a truly representative sample of test cases.

Consider test case *X*, which undergoes the following states throughout its execution cycle:

B→ *B*→ *F*→ *F*→ *F*→ *F*→ *P*

Now, check out test case *Y*, which has gone through the following states throughout its execution cycle:

F→ *P*

We can easily see that during test execution, test case *X* underwent different execution states, most of them in Failed status. We can also see that test case *Y* reached Passed status only in the second iteration of testing. Though it's not entirely wise to jump to the conclusion that there were more issues related to *X*'s execution than *Y*'s, this seems to be an assumption worth investigating. So, when we have to select test cases, such as *X* and *Y*, *X* will be the likely suspect for closer analysis.

From the above discussion, we can make the following observations:

- Test case states during a test case cycle, like many other defect metrics, are an important metric that should be captured. In some cases, a test management tool can be used to maintain a history of changes to the test cases during the execution, while in others, this step must be performed manually using Word and Excel documents.
- Test-case execution states can help you identify functionalities with inherent problems. For example, a test case that reaches the Passed state only after multiple state changes might be one to monitor for problems, compared to a test case that produced expected results during a first iteration.
- Information about test-case execution states can help you to focus on a representative sample of test cases out of thousands of test cases available. Once the test cases with inherent problems are identified, you can prioritize the test case to be executed in a test cycle.
- Test-case execution states, along with other factors, can help you make informed decisions about specific testing techniques, such as regression testing. The test-case execution metric can help answer the important question, "What to execute?"

Prioritizing from the Past

A history of test-case execution states is a useful metric that should be captured, logged and analyzed. An analysis of this history can help identify functionalities with inherent problems, assisting you in determining which to select for regression testing. It can also help you prioritize the test cases to be executed during the course of a product life cycle. Test-case execution state history has even greater value when a full regression suite of test cases must be represented by only a sample selection of test cases. With this vital tool, you can seek guidance from the past—your test cases' past, that is—to ensure your project's future success.

Still Buggy After All These Years

Things have changed seismically during Fran Schmidt's 16 years in the software industry.


Geoff Koch

First, there's the matter of titles. When she got her first job as a database librarian, the concept of a configuration manager wasn't "even a glimmer in someone's eye," says Schmidt, today a configuration manager with Source Medical in Birmingham, Ala.

Then there's the issue of who's doing the work. When Schmidt was beginning her career, it was a decidedly pink-collar pursuit, and she remembers it being rare to find a man in the field. "Now, I'm in the minority," she says.

Among the countless other changes are the explosion of gaming as a multibillion dollar market, the rise of government regulations and their associated reshaping of the way software is written, and the emergence of agile programming as an alternative to process-heavy approaches to building applications.

One constant, however, is that writing code was and is an inexact science. Defects still crop up on the way to the next release just as they did decades ago. And so the need for professionals proficient in defect tracking remains as urgent today as ever.

Though logging bugs can be as simple as sharing a spreadsheet or even a whiteboard, Schmidt cites the importance of using a defect tracker that's fully integrated with a source code control tool. Earlier in her career, trying to coordinate the parallel development tasks among three teams without the benefit of an integrated tool, Schmidt repeatedly became bogged down with lots of last-minute manual code tweaks.

Schmidt describes a typical scenario in that clunky project:

- Team A changes three files: foo1.cmd, foo2.cmd and foo3.cmd
- Team B changes two files: foo1.cmd and anotherfoo.cmd
- Team C changes two files: foo1.cmd and foo3.cmd

Now, say Team A requests a build. Since their changes are not the latest in the code stream, with many tools, including the one Schmidt was using, the configuration manager has to manually grab the correct version just to do the build. "There's a high chance of grabbing the wrong version under this scenario," she says, adding that the same danger exists when Team C requests a build.

The big problem comes when Team B requests a build. It's altogether too easy to fulfill the request and inadvertently lose Team C's changes. "Tools that aren't fully integrated won't catch this," Schmidt says.

Damon Poole, founder and chief technology officer at AccuRev, a Lexington, Mass.–based change management vendor, says the best way to ensure a tight integration is to use change packages, sometimes called *activities* and *merge tracking*. "Change packages give you a single high-level object that fully and simply defines a change, even if that definition has changed over time," Poole says.

The alternatives, he states, maintaining lists of files and versions or lists of change numbers, are unnecessarily manual and error-prone.

Defect-tracking tools may be the default when it comes to writing most office desktop or server software, but what about other thriving parts of the software landscape, such as games? After all, gaming coders and testers routinely collaborate with artists and writers to produce game titles. Might all this right-brain thinking mean that something simpler, like a whiteboard or spreadsheet, can suffice when it comes to collecting game development bugs?

No, says Sheri Pocilujko, the quality assurance chair for the Chicago area–based International Game Developers Association. "Games don't act like traditional software because there is often this open-endedness," she explains. "You now have to account for the limitless possibilities of what a player might try to do. This is exponentially increased in an online game, where you have multiple player interactions."

Spreadsheet-based defect tracking is often limited in its ability to support multiple users and provide reporting functionality—problems that loom just as large in game development as in the rest of the software industry. Besides, "with free or relatively free bug-tracking tools available on the Web, even an indie developer or new startup can afford to use a defect-tracking tool," says Pocilujko.

Wrestling with Regulations

Many established companies governed by strict, post-bubble regulations such as the 2002 Sarbanes-Oxley Act can't afford not to use a good defect-tracking tool. Crafted mainly to ensure accounting fidelity among U.S. publicly traded companies, Sarbanes-Oxley has affected more than the green eyeshade crowd. IT systems, so tied in most companies to the processing and reporting of financial data, must now be assessed for compliance with the Sarbanes-Oxley legislation.

Schmidt calls Sarbanes-Oxley, with its requirement that all changes be documented and audible, the single biggest change in how IT tools are selected in recent history.

"Your defect-tracking tool is the very heart of these audit trails," Schmidt says. "Having a tool that integrates to the source code is an absolute thing of beauty when your company is audited, as it allows the auditor to quickly and

Sixteen years ago, **Geoff Koch** was a recent high school graduate and believed he was defect-free. Time has proved otherwise. Write to him at koch.geoff@gmail.com.

easily see the change, its history and its result in a single place.

Sarbanes-Oxley's strictures apply only to U.S. public companies. Private firms can do defect tracking without fear of a financial audit. These firms, especially smaller and more entrepreneurial software-intensive shops, also may be more apt to embrace agile programming styles.

The Agile Advantage

By quashing bureaucracy and sclerotic business process, agile programming promises to produce cleaner code that does a better job solving customer problems. The now-famous Manifesto for Agile Software Development (agilemanifesto.org) espouses a value for "individuals and interactions over processes and tools," suggesting that, for agile adherents, a simple spreadsheet might be preferable to a more complex defect-tracking tool.

Lasse Koskela, a self-described XP/Scrum/Agile coach based in Helsinki, Finland, claims that agile teams often produce fewer defects, which in turn makes the tracking process that much easier. In the past year, Koskela, who works for Reaktor Innovations, has seen several projects go from using heavy-duty tools to tracking defects on a whiteboard adorned with sticky notes.

Koskela points to one project that was beginning to embrace agile processes in which using a whiteboard did wonders. Previously, defects had effectively been hidden from the developers, he says, and assigned out only by the project manager using the tool. Adding all defects above a certain priority/severity threshold to the whiteboard for all to see made it obvious what defects really needed to be fixed.

"One day, I realized that I was witnessing a [bug-fixing] competition between two developers," Koskela says. "That had never happened when the defects were just entries in an obscure tool.

"Visible information radiators are excellent tools," he continues. "While software is good at storing information, information radiators like whiteboards are good at making people act on that information."

Which isn't to say that the defect-tracking industry is evaporating anytime soon. Koskela acknowledges that there will always be cases where the whiteboard and spreadsheet will fall far short. And Schmidt says that much about programming today: increasingly complex development tools, models and architectures; teams distributed in multiple locations with minimal face-to-face communication; sped-up release cycles with resources moving in and out of multiple product versions faster than ever; and so on—these challenges all fuel the trend away from simplicity.

"All of this complexity adds up to one thing: the need for better organization," Schmidt says.

It also adds up to a likely steady stream of defects for Schmidt and her colleagues to track via whiteboard, spreadsheet, freeware or fully integrated tools, for another 16 years at least.

ALM—Where Do We Go From Here?


Rick Riccetti

Change is a constant of application life-cycle management (ALM). It's apparent everywhere as organizations continue to adopt new technologies, management methodologies and processes to remain competitive. By definition, ALM is a continuous process of tightly integrated steps at all stages of development: from planning and definition to development, quality testing and deployment. But ALM affects more than just the processes that control the whole software development life cycle. In many cases, it has a direct impact on other areas of the business. The future of ALM is driven by the goal of delivering business value and aligning with an overall business strategy.

We can learn much by examining the history of another sweeping technological change, that of customer relationship management (CRM) systems. Although CRM and ALM have little in common, they share an evolutionary path that began with departmental-level deployments and moved up through the enterprise as business value was gradually realized. As CRM evolved, point solutions were tied together to manage the life cycle of a customer from prospect through the end of the relationship. It was the integration of tools and subsequent availability of key performance indicators (KPI) such as customer activity, spending patterns and cost-of-service requests that enabled CRM to deliver real business value.

While CRM can be a boon to organizational efficiency, it hasn't always been a trouble-free journey. Aside from the substantial up-front cost of acquiring a CRM system, many companies found CRM frightfully expensive to deploy. A large part of the expense arose from the inflexibility of early CRM solutions. Organizations had to adapt their processes and workflow to the CRM system, rather than have CRM coexist with time-tested business processes. Integration of CRM with existing applications was also problematic due to inflexible architectures and proprietary interfaces. Until CRM vendors overcame these limitations with easier-to-use and more adaptable systems, CRM didn't become the crucial part of business strategy that it is today.

Compared to the mature state of CRM, application life-cycle management is still in its infancy. Coincidentally, ALM providers are following a path similar to CRM by integrating point solutions into seamless suites of functionality that include revision control, issue tracking systems, test management and automated testing tools. While this integration of tools is undoubtedly an important step, effective ALM requires more. The real business value of ALM will be unlocked when the tools improve operational performance to the extent that CRM has shown is possible.

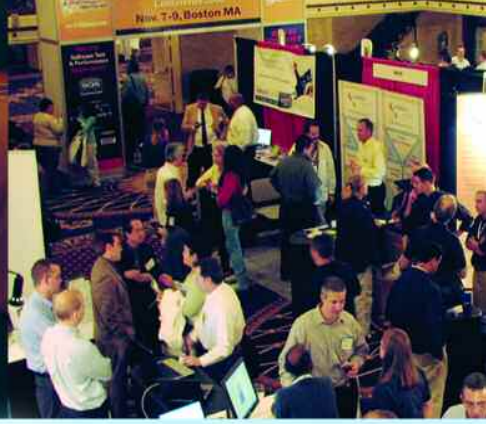
Current ALM solutions need refinement to truly help organizations better manage people, improve communications, reduce risk and aid with regulatory standards compliance and corporate governance initiatives. The future of ALM depends on developing relevant KPI that offer more help to management to measure the process of software development itself. Simply knowing how many defects a product has open is woefully inadequate. ALM must provide 360-degree visibility so the organization can easily determine what, if any, changes are required to increase the contribution of its development projects to operational performance. To achieve total visibility, every aspect of the development process from requirements definition to software design, code management and test verification must be tightly managed.

Although there is much to be gained from ALM, it has not yet achieved widespread and uniform adoption throughout the enterprise. In an organization working on a multitude of development projects, it is quite likely that teams use differing tools and development methodologies. Fully harnessing the power of ALM requires the enterprise to adopt at least some level of standardization so the things that ALM cares about (for example, tracking and managing change) can be done on an enterprise-wide basis. This also means that ALM vendors must produce highly flexible and adaptable tools that can be dropped into existing processes without disruption. Certainly, there's much to learn from the trials and tribulations of CRM.

The importance of software to modern business has never been greater. Many enterprises are de facto software development organizations no matter what products they're actually manufacturing or selling. Given this newfound criticality, an enterprise-wide ALM strategy is now central to business success. The future of ALM presents an opportunity to better exploit automation to reduce error-prone manual tasks, while improving operational performance across the organization. In the end, those organizations that adopt ALM will deliver faster-to-market, higher-quality software products and realize bottom-line advantages. ☒

Rick Riccetti is the CEO of Seapine Software in Mason, OH.


SQS
SOFTWARE QUALITY
SOLUTIONS
Solutions Without Limits™
Losing your grip implementing IT projects?
SQS can help you regain your footing.
Business Technology Optimization • IT Governance • Methodology
Performance Tuning & Optimization • Test Automation
Application Management • Quality Assessment • Product Training
www.SQS.com
MERCURY
ALLIANCE PROGRAM
PREMIER PARTNER
ORACLE PARTNER



Software Test & Performance CONFERENCE

November 7-9, 2006
The Hyatt Regency Cambridge
Boston, MA

Register Today for Early Bird Discounts! See Back Cover

Course Listing

SUPERB SPEAKERS!

Scott Barber
Rex Black
Clyneice Chaney
Ross Collard
Elfriede Dustin
Jeff Feldstein
Robert L. Galen
Robin F. Goldsmith
Hung Q. Nguyen
Robert Sabourin
Mary Sweeney
And dozens more!

TOTAL IMMERSION!

Choose From 67 Classes
Pick From 8 In-Depth Tutorials
NEW! Birds-of-a-Feather Sessions
Network With Colleagues
Ice Cream Social
Reception in Exhibit Hall
Pose Your Questions to 25+ Exhibitors
Mingle With More Than 40 Speakers

TERRIFIC TOPICS!

Managing Test/QA Teams
Testing SOA and Web Services
C# and ASP.NET Test
Security Testing
Locating Performance Bottlenecks
Just-in-Time Testing
Effective Metrics
Requirements Gathering
Improving Java Performance
Agile Testing
Risk-Based Testing Strategies
Test Automation

Produced by BZ Media

SD Times — SOFTWARE DEVELOPMENT — The Industry Newspaper for Software Development Managers

Software Test & Performance

Welcome

Save the Dates! November 7-9 at the Hyatt Regency Cambridge in Boston

From service-oriented architectures to AJAX to application security, change is the one constant in software development. In the face of continuous innovation, experts like you who are trying to improve the quality of your company's software encounter new challenges every day. And with code size and complexity increasing year by year, is it any wonder your information needs continue to grow as well? Now in its third year, BZ Media's Software Test & Performance Conference provides you with the practical, how-to information that will help you meet these challenges and make you successful in your profession.

The technical program for this conference was designed to serve the needs of people just like you: test and QA managers, development managers, test-focused developers and senior testers. The conference addresses such diverse topics as requirements management, security testing and test automation.

You can learn about testing service-oriented architectures and explore the fundamentals of database testing and how to recognize performance bottlenecks. Or you can delve into the intricacies of profiling J2EE applications, learn about performance tuning .NET applications and understand how to use metrics effectively to improve software quality. The three-day conference program packs in eight daylong tutorials plus more than 60 classes ranging from 60 to 90 minutes in length.

The faculty was handpicked for its technical expertise and ability to communicate. You'll meet and learn from industry luminaries like Rex Black, Rob Sabourin, Robin Goldsmith and Bob Galen. The program also features exciting keynote presentations to help give you a sense of where the industry is headed and what challenges you'll likely be facing next year.

While participating in the technical program is important, equally valuable is the opportunity you will have to meet with other software professionals outside the classroom. Conference activities are arranged to maximize your learning experience while leaving you time to compare notes with your classmates and confer with members of the faculty.

As an added bonus, the conference schedule and format will provide time for you to discover the latest products—which will be presented in the exhibit area—and pick the brains of the tool vendors. Read through the class listings and build a custom course of study over three days that will give you and your team tools and techniques that you can take back to the office and put into effect immediately. We look forward to seeing you at the Software Test & Performance Conference.

Register today at www.stpcon.com

Lindsey Vereen
Conference Chairman

NEW!

BIRDS OF A FEATHER SESSIONS

Join your colleagues and our instructors for informal discussions. Pose your toughest questions on the following break-out topics:

- **Test Automation**
- **.NET**
- **Testing for Security**
- **Java**
- **Agile Testing**
- **Improving Performance**

Tuesday, November 7
8:00 pm – 10:00 pm

Wednesday, November 8
8:30 pm – 10:30 pm

Note: Sign-up for these sessions will be on-site.

Contents

Event
Schedule

Monday, November 6

4:00 pm – 7:00 pm	Registration

Tuesday, November 7

7:30 am – 7:00 pm	Registration
8:00 am – 9:00 am	Continental Breakfast
9:00 am – 10:30 am	Full-Day Tutorials
10:30 am – 11:00 pm	Coffee Break
11:00 am – 12:30 pm	Full-Day Tutorials
12:30 pm – 1:45 pm	Lunch Break
1:45 pm – 3:15 pm	Full-Day Tutorials
3:15 pm – 3:45 pm	Coffee Break
3:45 pm – 5:00 pm	Full-Day Tutorials
8:00 pm – 10:00 pm	Birds of a Feather Session

Wednesday, November 8

7:30 am – 7:00 pm	Registration
7:30 am – 8:30 am	Continental Breakfast
8:30 am – 10:00 am	Technical Classes
10:00 am – 10:30 am	Coffee Break
10:30 am – 12:00 pm	Technical Classes
12:00 pm – 1:15 pm	Lunch Break
1:15 pm – 2:30 pm	Technical Classes
2:30 pm – 3:00 pm	Coffee Break
2:30 pm – 7:00 pm	**Exhibit Hall Open**
3:00 pm – 4:15 pm	Technical Classes
4:15 pm – 4:30 pm	Break
4:30 pm – 4:45 pm	Industry Keynote
4:45 pm – 5:30 pm	Keynote (Rex Black)
5:30 pm – 7:00 pm	**Reception in Exhibit Hall**
8:30 pm – 10:30 pm	Birds of a Feather Session

Thursday, November 9

7:30 am – 4:00 pm	Registration
7:30 am – 8:30 am	Continental Breakfast
8:30 am – 10:00 am	Technical Classes
10:00 am – 10:30 am	Coffee Break
10:30 am – 12:00 pm	Technical Classes
12:00 pm – 1:15 pm	Lunch Break
12:00 pm – 4:00 pm	**Exhibit Hall Open**
1:15 pm – 2:00 pm	**Ice Cream Social in Exhibit Hall**
2:00 pm – 3:15 pm	Technical Classes
3:15 pm – 3:45 pm	Coffee Break
3:45 pm – 5:00 pm	Technical Classes

Exhibit Hours:
Wednesday, 2:30 pm – 7:00 pm
Thursday, 12:00 pm – 4:00 pm

Register at www.stpcon.com

Updated June 6, 2006

Full-Day Tutorials

"This is the best conference I have attended. The instructors were extremely knowledgeable and helped me look at testing in a new way."
—Ann Schwerin, QA Analyst, Sunrise Senior Living

FULL-DAY TUTORIALS
Tuesday, Nov. 7
9:00 am – 5:00 pm

NEW

T-1 Assessing Your Test Team Effectiveness, Efficiency and More *By Rex Black*

As a test manager, you're probably looking for ways to do better and demonstrate your team's value and to find ways to improve it. This tutorial delivers. You'll learn techniques to assess your team that are driven by insightful questions and careful data analysis. By applying the ideas in this tutorial to each of the 12 critical testing processes, you'll know where you and your team stand.

This one-day tutorial is very hands-on with lecture primarily used to stimulate discussion. After each process is discussed, attendees will work through exercises that estimate performance metrics for their own test teams. After each exercise, attendees have a chance to discuss their results.

T-2 Testing Techniques: Theory and Application *By BJ Rollison*

This tutorial presents the formal theory and practical application of functional (behavioral) and structural (coverage) testing techniques. This tutorial will teach functional testing techniques, including exploratory testing, boundary value analysis, equivalence class partitioning and combinatorial analysis. Structural testing techniques covered include statement coverage, decision/branch coverage, condition and basis path coverage.

By attending this tutorial, you'll learn how to use functional testing techniques to establish a solid foundation and minimum baseline of test cases. You'll understand how structural testing techniques can be used to design additional tests from a white box approach to complement the test effort, to ensure that critical paths in the code have been exercised and to achieve higher code coverage results. You will also learn how to apply both black box and white box test design approaches to test more effectively.

T-3 Testing in Highly Iterative, Quasi-Agile Projects—Practical Strategies for Mixed Culture Projects *By Timothy D. Korson*

In the highly iterative, fast-paced environment of agile development projects, the traditional approaches to testing, quality assurance, requirements gathering and team interactions break down. QA managers trying to encourage best practices recommended by CMMI and SPICE find themselves at odds with developers trying to adopt best practices as recommended by the Agile Manifesto.

In the end, no one wins. Because of the constraints of corporate policies and management edicts, developers can't fully adopt agile practices. Because the developers do adopt as much of the agile process as they can get away with, the QA team finds that traditional approaches to quality management no longer work. Such projects must succeed in a "quasi-agile" development environment.

This tutorial will introduce you to software development processes and practices that affect your world. You will learn practical strategies for effectively integrating testing processes with modern software engineering processes. You will learn how to create effective tests, both component-level and system-level, for modern software systems. Detailed case studies will convey specific techniques for testing both components and entire systems.

T-4 Twenty-One Ways to Spot—and Fix—Requirements Errors Early *By Robin Goldsmith*

While many organizations have begun paying closer attention to defining requirements, few fully realize the need to know that their requirements are accurate and complete, and few know how to test requirements effectively. Most rely on one or two weak methods and have little awareness of how many errors they've missed—errors that later turn into expensive feature creep. This interactive class explains why it's so hard to test requirements, and it introduces 21 increasingly powerful methods to help you find frequently overlooked requirements errors when they are easiest and least expensive to fix.

Following the instructor's proven CAT-Scan Approach, participants apply the techniques successively to a real case and discover how each different method reveals additional, otherwise overlooked defects in the requirements. Participants learn ways to find previously overlooked requirements, increase meaningful customer/user involvement, enhance communications and understanding and test the adequacy of requirements definitions.

Development or Test/QA Managers' Classes

Special Track!

Need a higher-level view? You've come to the right place! Here are classes just for you—but feel free to deep-dive into any class or topic you want!

- **T-1** Assessing Your Test Team Effectiveness, Efficiency and More – **Black** (Full-Day Tutorial)
- **T-5** Foundations of GUI Test Automation – **Makedonov** (Full-Day Tutorial)
- **107** Designing for Testability – **Feldstein**
- **108** Testing the Software Architecture – **Sangal**
- **109** Effectively Training Your Offshore Test Team – **Hackett**
- **203** How to Turn Your Testing Team Into a High-Performance Organization – **Hackett**
- **207** Analyze the Return on Your Testing Investment – **Black**
- **209** Quality Throughout the Software Life Cycle – **Feldstein**
- **304** Managing Acceptance Testing Cycles More Efficiently – **Makedonov**
- **403** Deciding What Not to Test – **Sabourin**
- **502** Creating and Leading the High-Performance Test Organization, Part 1 – **Galen**
- **509** Recruiting, Hiring, Motivating and Retaining Top Testing Talent – **Feldstein**
- **602** Creating and Leading the High-Performance Test Organization, Part 2 – **Galen**
- **702** S-Curves and the Zero Bug Bounce: Plotting Your Way to More Effective Test Management – **Bradshaw**
- **705** Coding Standards and Unit Testing—Why Bother? – **Hendrick**
- **707** Performance Testing for Managers – **Barber**
- **801** Effective Metrics for Managing a Test Effort – **Bradshaw**
- **806** Best Practices for Managing Distributed Testing Teams – **Stevens**

"Great topics—well presented by reputable presenters. Having attended two years in a row, I have yet to be disappointed."
—Ardan Sharp, QA Manager, SunGard

NEW

T-5 Foundations of GUI Test Automation *By Yury Makedonov*

Testers and managers find themselves between a rock and a hard place when implementing test automation. From one side they are bombarded by a constant stream of sales pitches promoting the "click, click, click" record-and-replay approach. From the other side they are pressed by test automation gurus promoting their own frameworks. So, it's a challenge to keep their sanity under these conditions and to make sensible test automation decisions on tool and framework selection and test automation management.

In this real-world tutorial, major myths and misconceptions are dispelled, and explanations are provided on how to ensure the efficiency of GUI test automation. The tutorial covers major principles and current industry standards of GUI test automation, how to decide if a specific project should be automated or not, how to define a scope for test automation, how to select a test tool to automate a specific application and much more.

NEW

T-6 Software Endgames: How to Finish What You've Started *By Robert Galen*

We've all survived more than one software project that ended badly, where either the requirements were misunderstood or were implemented poorly. Or overall quality targets couldn't be met because there were simply too many defects. Or the team simply couldn't decide on priorities and in which direction to steer the project.

Many projects fail during testing. Not because of the testing per se, but because of the massive discovery of defects and functional gaps that indicate the true viability of the project. I call this time the Software Endgame, and I've spent a great deal of time negotiating its challenges through numerous software projects.

This presentation focuses on a set of high-level practices and techniques that will help improve your management and project steering within the endgame, providing guidance that will increase the odds of successfully delivering a project. You'll learn how to create an endgame delivery map that directs your release and testing milestones via entry/exit criteria; the importance of release criteria within the endgame and high-level rules of thumb for defining them; how to manage defect repairs—where to focus your efforts and scheduling rules of thumb—plus the many options you have for "fixing" defects.

T-7 Using Metrics to Improve Software Testing *By Alfred Sorkowitz*

Software metrics can aid in improving your organization's testing process by providing insight and early visibility into the "real" status of the testing effort and helping to make assessments as to whether progress, productivity and quality goals are being met. This tutorial presents a practical guide on how to start taking advantage of these new tools/techniques to aid in improving the testing process. These metric-based tools and techniques have successfully been used by software test teams, software developers and SQA/IV&V staffs.

In this class, you will receive an overview of software quality goals, criteria and metrics. You will also learn:

- The cost of inadequate software testing, a set of government/industry "best practices" metrics that can track the real status, quality and productivity of the testing effort, as well as provide an indication of future problems.
- Software complexity metrics: A new structured testing methodology that uses metrics to aid in developing software that is easier to test and maintain and selecting an appropriate set of paths for more thorough testing.
- How to integrate software metrics into the testing process.

T-8 Creating Agility and Effectiveness in Software Testing *By Linda Hayes*

NEW

Most companies resist test automation until the software is stable, reasoning that any savings from automation will be offset by the maintenance required to keep up with changes. Also, traditional record/script/replay approaches can't be implemented until the code is functional, which is too late in an agile development environment.

This session will present an incremental approach to test automation that supports an agile development environment by allowing automated tests to be written before the code and then be rapidly updated as changes are introduced. You will learn how to improve code development practices through automation, define executable requirements and write self-documenting automated tests before code is developed. You will also learn how to implement a test tool and platform agnostic automation architecture and automate test case maintenance for rapid response to changes.

TECHNICAL CLASSES
Wednesday, Nov. 8
8:30 am – 10:00 am

101 Seven Low-Overhead Software Process Improvement Methods *By Robin Goldsmith*

For many, software process improvement is synonymous with high-overhead, long-term, organizationwide initiatives that often are resisted and fail to produce the desired results. In this interactive presentation, you'll learn seven methods that can help you make software faster, cheaper and more reliable without all the hoopla. Key to meaningful results is recognizing, measuring and then specifically improving high-payback aspects of the instructor's proven REAL software process, which often differs considerably from what we presume we are doing. In truly agile fashion, applying these methods proficiently focuses efforts most efficiently on effectively producing useful software from the start.

102 Just-in-Time Testing Techniques and Tactics, Part 1 *By Robert Sabourin*

As the Boy Scout credo goes, "Be prepared." In this class, you will learn how to be ready for just about anything in a software testing project within the volatile environment of a Web or e-commerce software project. Managers will learn an array of techniques to manage and track software testing in chaotic environments—specifically, projects with continuously changing requirements and shifting priorities. Members of the development and testing teams will learn how, even while working with minimal information, to develop tests and converge the product development effort.

103 Automated Database Testing: Testing and Using Stored Procedures *By Mary Sweeney*

Today's heterogeneous data environments place an increasingly heavy burden on test engineers. Applications, whether Web-based or client/server, must be tested for seamless interface with the back-end databases; this typically goes far beyond what the popular test automation tools can provide. The intricate mix of client/server and Web-enabled database applications are extremely difficult to test productively. As a result, you are increasingly expected to know how to create and use SQL queries, stored procedures and other relational database objects to effectively test data-driven environments.

Technical Classes

"Good immersion in testing concerns."
—James Fields, Development Manager, Data-Vision Inc.

In this class, you will learn about testing at the database layer as an important adjunct to current tests. Using demonstrations and code examples, the instructor will present tips and techniques for creating efficient automated tests of the critical database back end using SQL, scripting languages and relational database objects. You'll learn why testing of database objects and stored procedures is necessary; how simple and effective automated tests for the back end can be created using various programming languages, including PERL and VBScript; and how to successfully test database objects, such as stored procedures and views, with many examples and code.

104 Lessons Learned in Test Automation, Part 1 *By Elfriede Dustin*

This class will present and discuss a series of automated testing lessons learned from actual experiences and feedback from real projects. You'll learn how to avoid some typical false starts and roadblocks when you implement your test automation efforts.

Part 1 of this class includes a discussion of better ways to define automation criteria, how to avoid duplicating the development effort when designing automated test cases, how to create reusable automated test cases, the need to verify all vendor claims in your own environment, the pitfalls of delegating the tool selection to a reseller or consultant and how to select the right tool. You'll also learn how to avoid losing sight of the testing efforts because developers or testers are too busy coming up with elaborate scripts to automate their unit and system tests.

Special Track!

Need To Tune Performance?

Crank up your applications' performance by choosing from these hot classes!

- **305** How to Optimize Your Web Testing Strategy – **Nguyen**
- **402** Accelerate Testing Cycles With Collaborative Performance Testing – **Cavallaro**
- **408** Techniques for Testing Packaged Application Performance – **Feaster**
- **503** Pinpointing and Exploiting Specific Performance Bottlenecks – **Barber**
- **504** Performance Tuning ASP.NET 2.0 Applications, Part 1 – **O'Mara**
- **603** SOA Performance Testing Challenges – **Barber**
- **604** Performance Tuning ASP.NET 2.0 Applications, Part 2 – **O'Mara**
- **608** Verifying Software Robustness – **Collard**
- **707** Performance Testing for Managers – **Barber**
- **803** Real-World Performance Testing Lab for (Almost) Free – **Flint**
- **805** Building a Bridge Between Functional Test Automation and Performance Testing – **Sody**

105 Code Coverage Metrics and How to Use Them *By Rex Black*

NEW

More and more testers and programmers are using tools that provide code coverage metrics. These metrics tell the tester or programmer how much of the code has been covered by a given set of tests and, more important, what conditions might not be covered. In addition, some tools can evaluate the coverage of data flows. Some tools can also provide insight into the complexity and, thus, the likely difficulty level of future refactoring of the code.

In this practical class, we'll examine the following code coverage metrics and how you can use them to write better code or tests: statement, branch, condition and loop coverage metrics; McCabe Cyclomatic complexity and basis path coverage; and data-flow coverage via set-use pairs. Each metric will be illustrated using a real program, with tests developed and run to achieve given levels of coverage. The code sample provided can be used, along with the course materials, to evaluate a given tool's ability to provide useful code coverage metrics.

106 Foundations of GUI Test Automation Using C#, Part 1 *By BJ Rollison*

NEW

There is a trend in the software industry for companies to seek candidates whose qualifications include the ability to write automated tests. Additionally, some companies realize the limitations of commercially available tool sets that use proprietary scripted languages. So, modern programming languages are being employed to develop more effective and more robust automated tests. But for testers who lack a programming background, the initial hurdle of learning a programming language can be a bit intimidating. This is further complicated by the fact that most programming courses teach us how to develop applications and not how to use a programming language to write automated tests.

This class is designed as a starting point for testers who lack a programming background, or for those with an understanding of programming concepts but are unfamiliar with automated testing using C# to test a Windows application. This class discusses common testing tasks illustrated with well-commented code examples and developed with free tools available on the Internet. In this session, you will learn how to launch, gather information, synchronize and close an application under test (AUT); manipulate and send test data to an AUT; and generate test data.

107 Designing for Testability *By Jeff Feldstein*

NEW

Many developers believe testing begins when all the features are complete and they hand off their work to the test team. While this may be what actually occurs in many development projects, it is far from the ideal. Software quality assurance begins in the definition phase of the project. One important aspect of software quality to consider is testability of the software architecture. Testability of your application can have a profound effect on its overall quality. This class will explain what to look for in a testable architecture, avoiding common mistakes and pitfalls, how to present your ideas to the development team and how to build test automation systems that take advantage of the testable architecture.

108 Testing the Software Architecture *By Neeraj Sangal*

NEW

Automated testing of the software architecture can keep quality from degrading and help preserve the design intent. In this class, a new lightweight approach to specifying and verifying the architecture is presented. Inter-module dependencies are utilized to represent the architecture of a software system using a dependency structure matrix (DSM). Once the architecture is specified, it can then be verified through automated tests during development. Furthermore, architectural violations can be easily prioritized for remediation. This approach will be presented through real life exam-

ples by applying it to a number of commonly used applications. Dependency analysis will be used to extract the architecture for applications such as Ant, JUnit, jEdit and the Eclipse platform. We will look at real examples of software development spanning several years to see how architecture evolves, how it often begins to erode and how regular testing can prevent this erosion.

109 Effectively Training Your Offshore Test Team *By Michael Hackett*

Working with offshore teams is a fact of life now for domestic test leads and managers, but many are still struggling to make their global test team work effectively. Training your offshore test team is critical to the success of your projects. If done right, training can help minimize your stress and late-night phone calls and ensure that you are getting the right information from the offshore team to make sure your testing effort is successful. Training of the offshore team needs to focus on a broad range of topics and must be specifically designed to the unique needs of that team.

In this class, we will discuss the key elements of successful offshore testing, including training in the areas of process, product/domain knowledge and testing techniques, and how training can be used as a retention tool for offshore staff. The class will include several real world examples based on the speaker's experiences working with teams in the most common offshoring locations.

Wednesday, Nov. 8
10:30 am – 12:00 pm

201 Prevent Showstopper Overruns With Risk-Based Proactive Testing *By Robin Goldsmith*

NEW

Project budget and schedule overruns frequently are caused by late, unplanned significant redesign and rework to fix showstopper errors. Traditional, reactive testing misses too many showstoppers or catches them too late to fix easily. In contrast, Proactive Testing's powerful risk analysis techniques identify many of the up to 75 percent of showstoppers that are ordinarily overlooked. Moreover, Proactive Testing can drive development to build systems in a truly agile different way that avoids much of the rework that showstoppers traditionally would have necessitated. The class will cover risk-based testing fundamentals and the limitations of traditional reactive testing approaches. You will learn how to continually refocus on testing higher-level risks more frequently and earlier in the testing cycle, as well as methods for identifying ordinarily overlooked showstoppers and reducing overruns.

202 Just-in-Time Testing Techniques and Tactics, Part 2 *By Robert Sabourin*

Please see description under Class 102.

203 How to Turn Your Testing Team Into a High-Performance Organization *By Michael Hackett*

All development managers, test managers and their organizations are looking for ways to improve quality. Quality improvement can come in many forms: reducing risks by delivering higher and predictable quality products; optimizing time-to-market; increasing productivity; and building a more manageable organization. Some managers look for quality improvement by attempting to implement a more standard or formal process. This sounds good, but where is the road map for how to get there? This class will help! You'll learn how to evaluate your test process and strategy, create a culture for change, implement change and use effective methods for measuring improvement.

204 Lessons Learned in Test Automation, Part 2 *By Elfriede Dustin*

We continue to explore automated testing lessons learned from actual experiences and from feedback based on real projects to help you to avoid some typical false starts and roadblocks when you implement test automation efforts. In Part 2 of this class, attendees will learn when automated testing doesn't speed up the testing effort. Attendees will also learn how to create mini development life cycles, how to maintain automated unit and system tests, how to implement smoke tests, and the pitfalls of using automated performance testing tools.

205 Database Security: How Vulnerable Is Your Data? *By Mary Sweeney*

There are many levels of software security. But how secure is the most important component of your application: your database? Quality control organizations must step up to the challenge of ensuring data security with appropriate tests that focus on this vital area. In this class, you will learn what the test team needs to know about protecting the server, the database connections, controlling access to database tables and restricting access to the database server itself. If data is in jeopardy, the entire system is at risk. We will also discuss the basics of security testing to ensure protection for the critical database component.

206 Foundations of GUI Test Automation Using C#, Part 2 *By BJ Rollison*

NEW

C# is a powerful programming language quickly becoming commonly used to develop test automation. But to effectively test Windows (Win32) applications written in C/C++, we must also learn to use common Win32 Application Programming Interfaces (APIs).

Part 2 of this class discusses how to use process invocation services to use Windows APIs to perform common testing tasks illustrated with well-commented code examples, and developed with free tools available on the Internet. In this session, attendees will learn how to:

- Import Win32 API library functions and marshal data types
- Get and set AUT focus
- Manipulate controls

"As a project manager, this conference fit my role well. Developers would also benefit."
—Lloyd Goss, Project Manager, JAARS

- Manipulate menus
- Create custom methods to perform repetitive tasks
- Create a reusable test library

207 Analyze the Return on Your Testing Investment *By Rex Black*

Testing is not just a good idea—it's a good investment. This class demonstrates the value of solid testing through quantifiable returns on the investment. Through effective developer testing, skilled development managers deliver solid, quantifiable benefits in four ways:

- Find bugs that get fixed—or even prevent them
- Find bugs that don't get fixed but are known
- Run tests that mitigate (potentially expensive) risks
- Guide the project with timely, accurate, credible information

Attendees will learn about all of those benefits—and how to measure them. A hands-on exercise teaches you how to estimate the return you could—or currently do—achieve on your test investment.

208 Using Scrum to Manage the Testing Effort *By Robert Galen*

Many testing efforts succumb to management and project pressures and become chaotic in terms of their focus and work quality. It's simply the nature of the endgame phase of software development projects, where anything goes in pushing for the delivery of a product, and it's usually quality that goes first. Beyond the product quality impacts, the team usually suffers, too, with low morale and little empowerment.

Scrum is one of the agile methodologies, and it focuses on project management in agile and iterative development efforts. It can be successfully applied to testing efforts to renew their focus and drastically improve overall results. In this presentation, we will explore the Scrum methodology and learn to apply it practically to your testing cycles. In this class, attendees will learn how the Scrum methodology applies to the testing effort.

209 Quality Throughout the Software Life Cycle *By Jeff Feldstein*

NEW

Software quality is everybody's job. Quality cannot be tested into the product; it must be emphasized, monitored and measured from the beginning of the project. Each team involved in the project, including product marketing managers, program managers, development engineering, documentation and test engineering, plays a key role in assuring software quality. A carefully planned application development life cycle is a key requirement to successful delivery of on-time quality software.

The application development life cycle consists of four broad phases: requirements, development, test and post-customer ship. Each phase has important activities that directly affect the quality of the delivered software. This presentation will explore each phase in detail from a software quality perspective. It will describe activities that need to happen at each step and the role of the test or software quality engineer, and will enumerate many common mistakes made. In addition you will learn how to catch bugs earlier in the life cycle when they are cheaper to fix.

Wednesday, Nov. 8
1:15 pm – 2:30 pm

301 Hacking 101: Donning the Black Hat to Best Protect Applications From Today's Hacking Threats *By Tom Stracener*

NEW

Applications have become fertile ground for attackers to uncover seemingly innocuous features and utilities in today's complex systems and gain unauthorized access. Hackers seek out weaknesses in the many modules and components of complex systems, looking for hidden fields, embedded passwords, exposed parameters to manipulate and ways to tinker with input strings and steal data. At a time when security is highly coveted, understanding the enemy's mindset is more crucial than ever. Therefore, one of the best defense mechanisms against hackers is to understand how they think—and recognize your network's Achilles' heels before a hacker exploits them.

In this class, you'll learn the thinking, strategies and methodologies commonly used by hackers and how to effectively implement a sound defensive plan that will help mitigate multiple attacks. Top Web application security flaws, including invalidated input, broken authentication and management, buffer overflows, injection flaws, insecure storage, denial of service and insecure configuration management, will be addressed so as to arm you with the right knowledge to protect your company's infrastructure.

302 Five Core Metrics to Guide Your Software Endgames *By Robert Galen*

By its very nature, the endgame of software projects is a hostile environment. Typical dynamics include tremendous release pressure, continuous bug and requirement discovery, exhausted development teams, frenzied project managers and long hours. Testing teams are usually in the thick of this battle and accustomed to these dynamics. However, project managers may not be proactive enough in working with their testing teams to understand the change and repair workflows within their projects. Yes, we work hard at managing bug reports, but we can do so much more to influence and focus a project's direction. In this presentation, attendees will learn how project managers can focus the entire team on a few key performance metrics to improve the overall endgame experience and increase the probability of delivering on time. And yes, to also survive yet another endgame.

"Best concentration of performance testing presentations/professionals I've seen."
—Nathan White, Manager, Testing Services, AG Edwards

303 Overcoming Requirements-Based Testing's Hidden Pitfalls *By Robin Goldsmith*

Testing based on requirements is a fundamental method that is relied on extensively. However, its thoroughness frequently can be compromised by traps that testers are not aware of. In this interactive presentation, you'll learn key sources of requirements-based testing oversights, including distinguishing business requirements from system requirements; assessing the extent to which the requirements are complete; the premise of one test per requirement; the appropriate level of test case detail; and developers' inclusion of requirements-based unit tests. The class will also focus on the strengths and often unrecognized weaknesses of requirements-based tests; the importance of testing based on business as well as system requirements and determining how many tests a requirement needs.

NEW 304 Managing Acceptance Testing Cycles More Efficiently *By Yuri Makedonov*

Once in a while a supposedly "almost completed" project requires one more cycle of acceptance testing and then just one more, then another and so on. Surprised management looks on in disbelief as the project spirals out of control down into a bottomless pit of "acceptance testing cycles." Yet, often management does not have a 100 percent clear and correct understanding of what is happening and why it's happening. As a result, their actions might not necessarily be very effective in controlling the situation. In this presentation, you will learn the different reasons for why these testing cycles can happen and specific techniques for getting a project like this out of a tailspin.

305 How to Optimize Your Web Testing Strategy *By Hung Q. Nguyen*

One of the key strategic challenges of Web testing is the dominance of change. Another key challenge is interdependence. Web applications are fundamentally dependent on cooperating tools and processes. Many of the processes, tools and standards in use by groups that do Web testing were originally developed with simpler and less dynamic situations in mind. Used by skilled and thoughtful people, in the context of a clear strategy, these processes and tools can add value. But if we allow them to drive our testing practices, they can easily do more harm than good. In this talk, you will learn how to analyze and optimize your Web testing strategy by selecting the right types of tests, how to execute them at the right time with a balanced number of cycles, and how to drive changes to improve your team's testing throughput.

NEW 306 Model-Based Testing for Java and Web-Based GUI Applications *By Jeff Feldstein*

Classic test automation simply repeats the same tests (with optionally varying data) until it stops failing or the application ships. The problem with this approach is that customers rarely flow through the application in the same sequence as the automation, and thus they are likely to find bugs that the automation missed. Model-based testing is a form of automated testing that brings random and flexible behavior to your automated test cases.

Model-based testing can be used for many types of software or application testing. This class will teach how to implement model-based testing, specifically as applied to Java and Web applications. Part of the course includes a demonstration of model-based testing; you will be able to download the XDE Tester source code used in the demonstration. Although the example application tested by this source code is fairly simple, it contains all of the data structures, concepts and program flow for implementing a large-scale, industrial strength, model-based test system.

NEW 307 Taking AIM—Using Visual Models for Test Case Design *By Robert Sabourin*

Designing test cases is a basic fundamental skill all testers master over time. This workshop teaches a fun graphical technique to help design powerful test cases and choose test data that will surface important bugs fast. These skills can be used in exploratory, agile or engineered contexts—anytime you need to design a test.

Mindmaps are powerful graphical tools used to help visualize complex paths and relationships between concepts. The workshop shows how Mindmaps can be used to visualize test designs and help understand variables being tested, alone and in complex combinations with other variables and conditions. The AIM (Application Input Memory) heuristic is taught through a series of interactive exercises. Real recent project examples are used to demonstrate these techniques. We will look at using some widely available free open-source tools to help implement great test cases and to help focus our testing on what matters and quickly hone in on critical bugs! If you are new to testing, these techniques will remove some of the mystery of good test case design. If you are a veteran tester, these techniques will sharpen your skills and give you some new test design approaches.

NEW 308 Elements of Software Design for Unit Testing *By Thierry Ciot*

Too often, software projects are built around classes and components that incorporate a tangled web of dependencies that can hinder—or even prevent—unit testing. What steps can you take to keep this headache out of your project? This course, for beginner to intermediate programmers/testers and managers, will introduce you to the design guidelines that facilitate unit testing. By building applications around classes and components with clearly defined dependencies, code will be more unit-testable by design. Following these guidelines can also greatly reduce the number of stubs or mock objects needed to perform unit testing. The guidelines presented in this course are based on solutions that have been used in actual projects, and each guideline is illustrated with code examples

All Sabourin, All the Time!

Special Track!

Rob Sabourin is one of our highest-rated speakers, and heck, he just loves teaching. If you like his workshop style and are ready to think out of the box, then don't miss these classes!

- **102** Just-in-Time Testing Techniques and Tactics, Part 1
- **202** Just-in-Time Testing Techniques and Tactics, Part 2
- **307** Taking AIM—Using Visual Models for Test Case Design
- **403** Deciding What Not to Test
- **507*** Unit Testing for Agile Development, Part 1
- **607*** Unit Testing for Agile Development, Part 2
- **708** What Hollywood Can Teach You About Software Testing

**Note that the Agile classes are limited to 30 participants, so register early!*

Register at www.stpcon.com

"I learned things I didn't know existed! I met people from all ranges of QA, all of whom were brimming with information they were willing to share."
—Rene Howard, Quality Assurance Analyst, IA Systems

that emphasize practical solutions. The class will end with a review of a complete real-world example.

Wednesday, Nov. 8
3:00 pm – 4:15 pm

NEW 401 The Secure Software Development Life Cycle *By Elfriede Dustin*

According to Gartner and Symantec, most business security vulnerabilities are now at the application layer. Attackers are focusing their efforts on regional targets, desktops and Web applications that potentially give attackers access to personal, financial or confidential information. Consequently, companies responsible for developing software must build security into their products as they are being developed.

This class focuses on application security throughout the software development life cycle. Attendees will learn secure coding guidelines that will help prevent defects from getting into code and that they must adhere to during the development process. They will learn about the Secure Software Development Life Cycle (SSDL) and its relationship to system development starting with the guidelines for security implementations.

The class covers the security program review and assessment activities that need to be conducted throughout the testing life cycle and the secure deployment considerations that have to be implemented. It also addresses the metrics and final review and assessment activities that need to be conducted to allow for adequate and informed decision making.

Special Track! Secure Your Software!

Securing the network is fine, but it's not enough! These classes will help you test your software for security.

- **205** Database Security: How Vulnerable Is Your Data? – **Sweeney**
- **301** Hacking 101: Donning the Black Hat to Best Protect Applications From Today's Hacking Threats – **Stracener**
- **401** The Secure Software Development Life Cycle – **Dustin**
- **606** Exploiting Web Application Code: The Methodologies and Automation of SQL Injection – **Fisher**
- **701** The Five Most Dangerous Application Security Vulnerabilities—And How to Test for Them – **Basirico**

402 Accelerate Testing Cycles With Collaborative Performance Testing *By Rick Cavallaro*

Testing and tuning the performance of enterprise Web applications is a complex task, undertaken by a team of individuals that may include performance engineers, QA testers, architects, developers, database administrators and related project team members. Fostering communication among these individuals can be challenging and can often lead to testing delays. The process is especially difficult when testers and developers are distributed around the building, around the country or even around the globe.

This session will provide a new methodology for collaborative load testing—an antidote to the iterative, multiweek process based on e-mail and conference calls that most organizations are forced to use today. You will learn:

- The drawbacks of traditional approaches to performance testing
- How to incorporate a team-based methodology for performance testing
- A new solution for collaborative load testing in a Web-based environment
- How outsourcing can impact QA efforts, and what you can do to mitigate that impact

NEW 403 Deciding What Not to Test *By Robert Sabourin*

Software project schedules are always tight. There is not enough time to complete planned testing. Do not stop just because the clock ran out. This presentation explores some practical and systematic approaches to organizing and triaging testing ideas. Testing ideas are influenced by risk and importance to your business. Information is coming at you from all angles—how can it be used to prioritize testing and focus on the test with the most value? Triage of testing ideas, assessing credibility and impact estimation can be used to help decide what to do when the going gets tough! Decide what not to test *on purpose*—not just because the clock ran out!

404 Putting the User Back in User Acceptance Testing *By Robin Goldsmith*

User acceptance testing (UAT) is often a source of consternation. Even though the process takes up considerable user time, too many defects continue to slip through, and users increasingly beg off from participating with claims that they don't have the time. Both effects may be symptoms of professional testers' mistaken conventional wisdom about the nature and structure of UAT. In this eye-opening presentation, you'll learn ways to gain user confidence, competence and cooperation. Plus, you'll learn how to create user-driven UAT that increases user testing competence and confidence.

405 Getting a Handle on Risk: Risk-Based Testing Strategies *By Clyneice Chaney*

With the rapid pace of application development, testing has become a challenging proposition. Trying to meet tight deadlines and deliver products that meet customer requirements is the greatest challenge testers face today. This presentation discusses a risk assessment tool that is used to assess risks associated with product testing. The assessment tool provides an alternative to "guesses" about what should be tested and helps test managers determine where they should concentrate their efforts.

The proposed risk strategy for testing moves us from the informal approach experienced testers often use to a more formal and systematic way of assessing risk that allows you to base your test strategy on the assessment as well as address the quality concerns of the stakeholder.

NEW 406 Java EE Performance Tuning Methodology: Wait-Based Tuning *By Steven Haines*

Java EE performance experts know that the key to success is to focus application tuning effort where it's needed most—at the wait-points, where the delays happen. But what are the wait-points, how can you find them in your application, and what can you do about them? That's what you'll learn in this intermediate-level class.

Wait-points can encompass Web and business tier thread pools,

Keynote Address: Rex Black • President, RBCS
Wednesday, November 8 • 4:45 pm – 5:30 pm

Five Trends in Software Engineering

Five strong winds of change are blowing in the software and systems engineering world. As winds affect a sailboat, these winds of change will affect software engineering, including development and testing, as a field, and software engineers as a community. Your career is at stake, and both risks and opportunities abound. In this talk, Rex Black will speak about these five trends and how they affect software engineering. He will offer cautions about the risks and identify the potential opportunities you face. For each trend, he will provide references to books and other resources you can use to prepare yourself to sail the ship of your software engineering career to the destination you desire: professional success.

external dependency connection pools, persistence caches, object pools and even garbage collection. We'll show you how to disassemble your application call stack, identify wait-points and tune from the inside-out to optimize throughput and suspend requests where they are best suited to wait. You'll leave this class knowing how to focus your application tuning efforts to immediately improve the performance of your Java EE systems.

NEW 407 Agile Test Development
By Hans Buwalda

Agile methods have become standard in the software development world. The emphasis on short, iterative cycles, constant feedback and a team-based approach to quality has proven effective for delivering software on time and on budget. The same approach can be applied to developing your tests and test automation, even if your development project is using a traditional "waterfall" life cycle. Good test design, especially good automated test design, requires constant feedback from project stakeholders outside the QA team, including the development team, management and customers. The tests should go through several "iterations" of review before being put into "production" against the system under test.

This class will discuss an agile approach to building tests and test automation, so that the QA team can ensure that the system is tested early and often, thereby taking testing off the critical path to releasing the product. This class will present a methodology and case study to illustrate how agile test development can be implemented in real world projects.

NEW 408 Techniques for Testing Packaged Application Performance *By Michel Feaster*

Enterprises are investing more and more in ERP/CRP applications like SAP and Oracle. While these are more critical to the business than ever, new paradigms such as SOA, agile development and offshoring are creating an added layer of complexity for QA organizations. This presentation will drill down on performance testing techniques, including workload modeling, endurance and stress testing, diagnosis and problem isolation and automated script creation. In addition, the speaker will address how to create an optimal team structure and enhanced reporting and communications techniques.

Thursday, Nov. 9
8:30 am – 10:00 am

501 Strategies and Tactics for Global Test Automation, Part 1 *By Hung Q. Nguyen*

We automate software testing to gain speed. We organize our distributed teams globally to maximize round-the-clock coverage and cost efficiency. Both solutions fulfill legitimate objectives. However, implementing them successfully while keeping the risks contained with a high degree of certainty proves to be an enormous challenge. In this class, through a series of technical and management case studies and real life examples, you will learn about seven steps that will deliver return on investment through a global test automation program. Attendees will learn how to: assess testing strategy and needs; minimize the costs and risks of global resources; select the right test automation technology for the job; align testing with business processes and development practices; and measure, analyze and optimize for continuing improvement.

NEW 502 Creating and Leading the High-Performance Test Organization, Part 1
By Robert Galen

Issues such as fewer people, less time, constantly changing technologies and increasing business expectations are clearly the norms for what software teams must face today. Nowhere is this more evident than within testing teams, since the pressure increases as we move down through the life cycle. This pressure poses a tremendous leadership challenge for testing team managers, group leaders or anyone chartered with directing testing. However, this challenge creates the opportunity for effective test leaders to differentiate themselves and their teams as they meet and exceed organizational expectations.

This two-part class focuses on acquiring the fundamental skills to *become* that outstanding test leader. It will explore such issues as how to: build, motivate and lead great testing teams; create impact driven communications on testing state; properly plan and execute your team's evolution, growth and ability to meet project challenges; handle the toughest "people" challenges facing good managers; and be agile and adaptable—learning to change with the organizational landscape.

503 Pinpointing and Exploiting Specific Performance Bottlenecks *By Scott Barber*

One part of the system is always slowest—the bottleneck. Until you remedy that bottleneck, no other tuning will improve performance along that usage path, but before you can tune it, you must first conclusively identify it. Once the bottleneck has been identified, the resolution can be reached more quickly if you modify your existing tests to eliminate distraction from ancillary issues. Pinpointing the bottleneck precisely is an art all its own.

Designed for technical performance testers and developers/architects, this class will show how the performance testing team and the development team can work collaboratively to analyze results and identify bottlenecks by tier, component and object. Then you'll learn how to design tests to exploit those bottlenecks for tuning purposes with examples using IBM Rational and free tools.

504 Performance Tuning ASP.NET 2.0 Applications, Part 1 *By Thomas O'Mara*

This class will provide an intuitive understanding of how to set up a solid testing infrastructure, will help you gain an in-depth understanding of critical .NET and ASP.NET components, and will show you how to monitor the operating system and the ASP.NET application in real time. You will also get a good overview of statistical measurements and their meanings as applied to the data.

The first part of this class will take an in-depth look at the .NET Framework 2.0 that includes the Common Language Runtime, Windows 2003 Server with Internet Information Server 6.0 and ASP.NET 2.0 architecture as it relates to performance tuning.

Part 2 of this class will offer an in-depth real time look at the critical performance counters on Windows 2003 Server, IIS 6.0 and ASP.NET 2.0 as they provide feedback on the health of the application running under load.

505 Identify and Mitigate Risks Through Testing, Part 1 *By Rex Black*

Over the past 10 years, professionals working in software and sys-

"I've received volumes of new information and ideas to share with my team."
—Theresa Harmon, Business Applications Developer, Pharmacare Specialty Pharmacy

tem development have learned how to apply the powerful techniques of risk analysis and risk management to their projects. In this class, you will learn:

- How to apply risk analysis techniques ranging from informal discussions to ISO 9126 to failure mode and effect analysis
- How risk prioritization can tell system development professionals where to focus development and test resources
- How the project team can improve the accuracy of the risk analysis—and thus the effectiveness and efficiency of testing—throughout the system development life cycle

Part 1 of this class will discuss the various techniques and illustrate them through real case studies. Part 2 includes a hands-on exercise to prepare you to apply these powerful techniques to your next project.

506 Rapid Business-Driven Testing *By Clyneice Chaney*

Structured testing is a vital part of any development project. The problem is that almost no one is given the time and resources to properly execute a thorough test process. In an ideal world, rapid testing would not be necessary, but with most development projects there are schedule crunches and times when a quick assessment of the product quality is necessary.

Rapid testing is a way to scale thorough testing methods to fit arbitrarily compressed schedules. "Rapid" doesn't mean "not thorough," but it does mean as thorough as is reasonable given constraints on time. In this class, you will learn how to use new rapid business-driven testing techniques, methods and templates that will increase product quality in rapid development projects.

507 Unit Testing for Agile Development, Part 1 *By Robert Sabourin*

With the increasing popularity of agile development methods, the role of testing is starting earlier in the software development cycle. Testers and developers are challenged to develop software at lightning speed, often using new and untested technologies. The class will show you how development and testing teams can work together to promote and implement improved unit testing. Attendees will learn how to save your company money by finding and fixing bugs long before system testing even starts. Get the ammunition you need to convince management of the economic and business benefits of comprehensive unit testing.

This two-part class addresses unit testing issues within the context of different development life-cycle models, especially new agile approaches, and demonstrates the tools and techniques needed to organize for and implement unit testing. The class is taught in workshop style and includes many hands-on group and team exercises, examples and unit testing tool demonstrations.

Due to the interactive nature of these workshops, class size is limited to 30 people.

508 Using Code Metrics for Targeted Code Refactoring *By Andrew Glover*

Oftentimes, candidate code for refactoring is based on subjective determinations. The proper uses of code metrics, such as cyclomatic complexity, fan-in, fan-out and depth of inheritance, can also facilitate the discovery of candidate code that is in need of refactoring. For example, cyclomatic complexity is adept at spotting methods containing a high degree of conditional logic, which, consequently, can be replaced with polymorphism as elaborated by Martin Fowler. Additionally, excessively deep hierarchy trees create problematic testing targets, which can be broken out into separate objects with Fowler's Replace Inheritance with Delegation and Collapse Hierarchy patterns. Fan-in and fan-out are quite effective at pinpointing brittle code, which can be refactored into a more stable state with numerous patterns, including Extract Hierarchy and Extract Class.

You'll leave this class with an understanding of seven industry-standard code metrics; moreover, you will have the ability to utilize these metrics to spot "complex" code and will have a grab bag of techniques with which to improve the code.

509 Recruiting, Hiring, Motivating and Retaining Top Testing Talent *By Jeff Feldstein*

The expectations today are for increasingly high-quality software, requiring more sophisticated automation in testing. Test and QA teams must work more closely with development to ensure that this sophisticated automation is possible. This has led to software engineers applying creativity, talent and expertise to not just application development, but testing as well. This transition from manual to scripting to highly engineered test automation changes the way we recruit, hire, motivate and retain great test engineering talent.

The speaker uses examples of how his team at Cisco changed the way it tests over the past six years. In this class, he'll review eight reasons why test is a better place for software developers than software development, and he'll show how and when to express these points to hire, motivate and retain top talent. You'll see how to inspire greater innovation and creativity in your testing processes and how to manage and inspire test and development teams that are spread across different locations. You'll also learn the place of manual testing in the new environment.

Special Track!

Black is Back!

Rex Black was one of our most popular speakers at the first STPCon, and he's back. Sign up for one or more of these classes, and you won't be disappointed!

T-1	Assessing Your Test Team Effectiveness, Efficiency and More (Full-Day Tutorial)
105	Code Coverage Metrics and How to Use Them
207	Analyze the Return on Your Testing Investment
505	Identify and Mitigate Risks Through Testing, Part 1
605	Identify and Mitigate Risks Through Testing, Part 2

Thursday, Nov. 9
10:30 am – 12:00 pm

601 Strategies and Tactics for Global Test Automation, Part 2 *By Hung Q. Nguyen*

Please see description under Class 501.

NEW

602 Creating and Leading the High-Performance Test Organization, Part 2 *By Robert Galen*

Please see description under Class 502.

"This conference helps testers and developers as well as managers and leaders. There is enough variety and content for everybody."
—Michael Farrugia, Software Engineer, Air Malta

NEW **603 SOA Performance Testing Challenges** *By Scott Barber*

Officially, SOA stands for service-oriented architecture, though Martin Fowler quips that maybe service-oriented ambiguity would be more apropos due to the diversity of technical methods being used to implement the SOA concept. The great thing about this ambiguity is that while the developers, architects, vendors and standards groups struggle to narrow the range of technologies, we testers have a chance to get ahead in our preparations for testing these applications. This presentation will introduce the core concepts of SOA, discuss the challenges these concepts present to performance testing and finally map out a performance testing strategy that allows us to use SOA as a springboard to move the state of performance testing significantly forward in your organization and the software industry as a whole.

604 Performance Tuning ASP.NET 2.0 Applications, Part 2 *By Thomas O'Mara*

Please see description under Class 504.

605 Identify and Mitigate Risks Through Testing, Part 2 *By Rex Black*

Please see description under Class 505.

606 Exploiting Web Application Code: The Methodologies and Automation of SQL Injection *By Matthew Fisher*

SQL injection is a technique for exploiting Web applications that use client-supplied data in SQL queries without stripping potentially harmful characters first. Despite being remarkably simple to protect against, there are an astonishing number of production systems connected to the Internet that are vulnerable to this type of attack, due to the simple fact of improper input validation. Developers and quality assurance professionals who design, build and test business-enabling applications generally lack the security knowledge necessary to avoid creating common defects that are so easily exploited by hackers.

In this class, you'll learn about the techniques that can be used to take advantage of a Web application that is vulnerable to SQL injection. The session addresses proper mechanisms that should be put in place to protect against SQL injection, as well as overall improper-input validation issues.

607 Unit Testing for Agile Development, Part 2 *By Robert Sabourin*

Please see description under Class 507.

608 Verifying Software Robustness *By Ross Collard*

Do you like breaking things? If so, this session's for you!

It's not enough to design systems for dependability; we have to verify that reliability as well. Software is robust if it can tolerate such problems as unanticipated events, invalid inputs, corrupted internally stored data, improper uses by system operators, unavailable databases, stress overloads and so on. Systems that include both hardware and software are robust if they can tolerate physical problems such as equipment damage, loss of power and software crashes. Since these problems can and do occur in live operation, this session examines how to evaluate a system's robustness within the relative sanctity of the test lab.

609 Metrics: How to Track Things That Matter *By Clyneice Chaney*

Metrics programs have often been a dirty word, misused and poorly implemented. This class discusses ways to provide metrics that really matter to organizations and provide visibility into their or their customers' organizations. The class will begin with discussions about why metrics programs fail and will move on to discuss keys to successful metrics programs, developing quality metrics that matter and ways to implement and maintain these metrics over time.

Thursday, Nov. 9
2:00 pm – 3:15 pm

NEW **701 The Five Most Dangerous Application Security Vulnerabilities—and How to Test for Them** *By Joe Basirico*

The most difficult problems of IT security are found at the application layer. Exploitability of applications due to poor design has reached epidemic levels. Perimeter/network defenses are not enough to protect organizations from attacks and most software teams possess neither the tools nor the expertise to properly secure their applications. This class highlights the top five security vulnerabilities that face testers today. You will learn practical how-to tips for testing your applications with a security mindset to attack their applications before a hacker does.

NEW **702 S-Curves and the Zero Bug Bounce: Plotting Your Way to More Effective Test Management** *By Shaun Bradshaw*

The use of objective test metrics is an important step toward improving the ability to effectively manage any test effort. Two significant test metrics concepts, the S-Curve and Zero Bug Bounce, allow test leads and test managers to easily track the progress of a test effort, improve the ability to communicate test results and test needs to the project team, and make better decisions regarding when an application is ready to be delivered.

You will learn:

- How to establish an S-Curve: What an S-Curve is, why it is important in testing, how to develop a theoretical S-Curve, what metrics can be tracked using the S-Curve and how to track them.
- How to create a Zero Bug Bounce: What the Zero Bug Bounce (ZBB) is, why it is important in tracking defects, and how to generate a ZBB.
- Interpreting the graphs: Finally, we discuss a method for examining and interpreting the graphs created by these metrics to make improvements to the current test efforts as well as future development efforts.

NEW **703 Testing Java Programs—Memory Management Issues** *By Averil Meehan*

It is a myth that Java's garbage collection has solved memory management problems. These can still cause space leaks severe enough to crash a program or for a thread of execution to stop. Unfortunately, often such errors occur only when a program runs for a long time, so they may not show up until the testing stage of development. Such errors are extremely difficult to debug as the error can appear in one section of code yet be caused by another. The nature of memory errors means that the problem can manifest itself in different ways and at different times in code execution, something that also makes detection and solution problematic. In this class, we will discuss what memory problems can occur and consider different approaches to detecting the code that caused them.

Technical Classes

"Reputable speakers and presenters. Great class topics."
—Jung Manson, QA Manager, Webloyalty.com

NEW **704 Defining Test Data and Data-Centric Application Testing** ***By Chris Hetzler***

As applications grow larger and more complex and as automated testing of these applications is increasingly adopted, the data that is used during the execution of the automated tests needs to be clearly defined and identified early in the development life cycle. Currently, the Software Engineering Body of Knowledge does not contain a definition for "test data," nor do any of the top books on the subject of software testing. This presentation will propose a definition for the term "test data" and outline what testers can do to ensure that teams are considering it in their planning phases and provide useful ideas on how to isolate what those data needs might be. The presentation will use examples of software development projects that have defined their test data early and those that have not to provide you with context and examples.

705 Coding Standards and Unit Testing—Why Bother? ***By Joshua Hendrick***

Many developers think that the industry "best practices" of coding standards and unit testing are a waste of time: They require additional effort, but they don't seem to make your life any easier, or your code any better. This is not surprising.

This class explains how developers can apply coding standards and unit testing to improve their code and prevent the number of problems they need to identify, diagnose and fix over the course of the project. The first half teaches you how to apply coding standards to prevent errors related to code functionality, security and performance. The second part focuses on how you can extend traditional unit testing to expose reliability problems that could lead to instability, unexpected results or even crashes or security vulnerabilities. Discussion will include how these test cases can be leveraged to build a projectwide automated regression system that runs in the background each night and immediately alerts the team when code modifications or additions break previously verified functionality.

NEW **706 The 5% Challenges of Test Automation** ***By Hans Buwalda***

Some of the most common problems with software test automation projects are that too much effort is spent on developing test scripts where the percentage of tests that is actually automated is only a meager 20% to 30% because too much time is spent on script maintenance. These issues cause numerous pains, including skyrocketing costs, a lack of management visibility and questionable quality of tests.

These universal problems in test automation have led this speaker to introduce the 5% challenges of test automation: No more than 5% of tests should be executed manually, and no more than 5% of the test effort should be spent creating automation. The speaker will present the keys to meeting the 5% challenges and illustrate them through a case study of a successful project that maximized test automation while minimizing the time spent automating tests.

707 Performance Testing for Managers
By Scott Barber

Performance testing as an activity is widely misunderstood, particularly by managers and others not directly involved in doing it. This presentation details the most critical things for managers to know about the performance testing process and ways to improve it. Learning, understanding and applying these nuggets of knowledge to your current or future performance testing projects will dramatically increase your team's chances of success. In this class you will learn how to work with experienced performance testers to get the results you need, why performance testing should begin well before the application is fully functional and how to do it, and ways to better integrate performance-testing personnel into the development team effort. In addition, you will learn how to recognize the difference between "delivery" and "done" as they relate to performance testing and how to assess and balance the risks inherent in each. Also covered is how to create and maintain a program that will ensure not only that your performance testers have the tools they need, but that they will know how to use them and when to put them away.

NEW **708 What Hollywood Can Teach You About Software Testing**
By Robert Sabourin

Powerful lessons can be learned from some of the great epic movies of our day: "Star Wars" and bug triage, "Indiana Jones" and requirements, "Monty Python" and configuration management, "Jurassic Park" and unit testing, "The Usual Suspects" and teamwork, and "Star Trek" and SLAs.

There are important metaphors within these movie stories that you can apply to real test management problems. Robert Sabourin's entertaining talk ties practical real-world experiences to lessons learned from the movies, offers tips to manage your team and provides his unique insight into how to get things done.

You will learn:

- How to identify your testing project genre—epic saga, action thriller, mystery or comedy
- How to storyboard a testing project
- How to draw lessons from an unlikely source

Thursday, Nov. 9
3:45 pm – 5:00 pm

NEW **801 Effective Metrics for Managing a Test Effort** ***By Shaun Bradshaw***

When managing a test effort, test leads and test managers sometimes find it difficult to empirically convey the impacts of scope changes, delays and defects to upper management. This class introduces a set of well-defined test metrics related to tracking and managing a testing effort. It demonstrates how to consistently apply these metrics to software projects and how to improve communication of your findings to the rest of the organization.

The following concepts will be taught as a part of this class:

- Test Metrics Philosophy – A four-step strategy for creating, tracking and interpreting test metrics
- Base Test Metrics – Fourteen simple metrics easily tracked by test analysts that can be used to derive more sophisticated test management information
- Management Test Metrics – Ten metrics and two graphs that make managing and communicating the status of a test effort easier for test managers and test leads
- Interpreting Test Metrics – A method for examining and interpreting the data to make improvements to the current test effort as well as future development efforts

NEW **802 Software Testing a Service-Oriented Architecture** ***By Ted Rivera and Scott Will***

The presence of service-oriented architectures (SOA) has grown significantly in the past few years, and test professionals must stay abreast of the latest technologies in order to continue providing sig-

nificant value-add to the software development process.

This class describes briefly what SOA is, methodologies that can be employed when testing in an SOA environment, common and potential pitfalls that can be avoided, a comparison and contrast of testing in an SOA environment versus a "traditional" test environment and a look ahead at future opportunities.

NEW 803 Real-World Performance Testing Lab for (Almost) Free *By Aaron Flint*

This course will discuss how to set up and maintain an effective performance test lab without spending a great deal of money on specialized performance testing hardware or software. Topics that will be covered include: what is needed to set up an effective performance lab; the hardware needed to get the maximum performance from the lab; and the software that must be installed and run in the lab, including tuning options and software configurations. We will also discuss how to manage and run performance tests and generate results. Finally, we will talk about maintenance issues such as how to preserve the environment.

804 Testing Java Applications Using the Eclipse Test and Performance Tools Platform (TPTP) *By Paul Slauenwhite*

The Eclipse Test and Performance Tools Platform (TPTP) provides a flexible and extensible framework for creating and managing tests, deployments, datapools, execution histories and reports with extensions for performance, JUnit, GUI and manual testing of Java applications. In this class, you will learn how to use the performance, JUnit, GUI and manual test tooling in the TPTP for testing Java applications. Intended for developers and testers who want to test their Java applications, the class begins by providing an overview of the motivation, history and architecture for the TPTP project. Then the TPTP test framework is explained with a focus on extending the framework for vendor-specific purposes. Finally, the performance, JUnit, GUI and manual test tooling in the TPTP project is demonstrated to illustrate the life cycle of an application's test assets. Attendees are provided with the sample configuration and code used in the technical class.

NEW 805 Building a Bridge Between Functional Test Automation and Performance Testing *By Peter Sody*

In a lot of software development organizations there is a clear distinction between functional test automation and performance testing. This often leads to different teams testing either the functionality or performance of the same software systems. This class highlights the commonalities between functional test automation and performance testing and shows how these two areas can benefit from each other.

This class gives practical examples on how to benefit from a collaboration, covering aspects of the whole test cycle. Applied in the right way, this can go far beyond reusing common artifacts. With the right approach the teams can complement their skill sets and significantly increase the efficiency and coverage of their testing. You will learn how to determine the common areas that can boost the testing performance for both sides and how to recognize the limits of such efforts.

NEW 806 Best Practices for Managing Distributed Testing Teams *By Dean Stevens*

Software testing projects are now often completed using a distributed model to leverage offshore labs to lower costs or decrease project delivery time. But without building certain cultural, methodological and technological foundations into your organization's core values, it can be difficult or impossible to effectively manage projects being completed by dispersed virtual teams.

Managing a distributed testing project requires a highly disciplined approach. Developing an effective framework for the execution of distributed projects goes hand in hand with requiring consistently documented processes and standards ("best practices"). However, by architecting a work model to accommodate a distributed model, you ensure a rigorous approach that will assist the successful execution of projects according to a well-defined set of best practices, regardless of the execution model employed.

Once methodological and technical frameworks are in place, the bulk of the remaining challenges are related to "soft issues," including cultural incompatibility, leadership problems, trust issues and negative competitiveness. These are actually the major obstacles to successful completion of distributed projects. However, there are concrete ways to alleviate these problems, including redefining your corporate culture, improving project planning and adjusting project staffing and technical infrastructure. In this session, attendees will learn methodologies and best practices designed to build "teamness" between distributed groups and set up both the physical and project structure so your teams can succeed.

NEW 807 Diagnosing and Resolving J2EE Application Issues Before Deployment *By Ferhan Kilical and Stanley Au Yeung*

J2EE offers many advantages to developers but introduces many challenges to the development, performance diagnosis, tuning, deployment and management of applications in the enterprise network. In this class, attendees will learn how to proactively detect problems and isolate them at different layers and tiers. They will acquire the tools to be able to pinpoint root causes to specific application components.

The class begins with the end-user business process, then drills down to application layers, communication between different layers, transaction trend-request data, method calls and class data. During the class, the presenters will share some of the hands-on samples that they mastered during their own J2EE tuning efforts.

Conference and Tutorial Faculty

Scott Barber is software test manager at AuthenTec and a member of the technical advisory board for Stanley Reid Consulting. Previously, he was a consultant who focused on teaching and performing practical performance testing and analysis. His project-level experience was evenly split between testing and analyzing performance for complex systems and mentoring organizations in the development of customized corporate methodologies based on his performance testing approach.

Mr. Barber has a master's degree in IT from American Intercontinental University. He writes Peak Performance, the performance testing column in Software Test & Performance magazine, and he also speaks at many technical conferences.

Joe Basirico has spent most of his educational and professional career studying security and developing tools that assist in the discovery of security vulnerabilities and general application problems. His primary responsibility at Security Innovation is to deliver the company's Security Training Curriculum to software teams in need of application security expertise. Mr. Basirico has trained developers and testers from numerous world-class organizations, such as Microsoft, HP, EMC, Symantec and ING. He holds a B.S in computer science from Montana State University.

A 20-year-plus software and systems engineering veteran, ***Rex Black*** is president and principal consultant of RBCS, which offers training, assessment, consulting, staff augmentation, insourcing, offsite and offshore outsourcing, test automation and quality assurance services. Mr. Black has published several books, including "Managing the Testing Process" and "Critical Testing Processes." He has also written more than 20 articles, presented hundreds of papers, workshops and seminars, and given more than a dozen keynote speeches at conferences and events around the world. Mr. Black is the president of both the International Software Testing Qualifications Board and the American Software Testing Qualifications Board.

Shaun Bradshaw serves as director of quality solutions for Questcon Technologies. He is responsible for managing Questcon's team of senior practice managers in the areas of quality solutions development and service delivery, and also works with clients to improve their quality assurance and software test processes.

Hans Buwalda leads LogiGear's action-based testing (ABT) research and development, and oversees the practice of ABT methodology. Mr. Buwalda is an internationally recognized expert specializing in action-based test automation, test development and testing technology management. He's also a speaker at international conferences, delivering tutorials and workshops, as well as presenting testing concepts such as ABT, the three Holy Grails of test development, soap-opera testing and testing in the cold. Recently, Mr. Buwalda co-authored "Integrated Test Design and Automation." He holds an M.S. in computer science from Free University, Amsterdam.

In his five years at Empirix, ***Rick Cavallaro,*** senior applications engineer, has worked with hundreds of companies helping to ensure the performance of their most critical Web applications. A 10-year veteran of the software industry, he specializes in testing and application development. Prior to joining Empirix, Mr. Cavallaro served in engineering roles at Aviv, Workstation Solutions and Revelation Software. He holds a BSEE degree from the University of Massachusetts, Lowell.

Clyneice Chaney, quality manager at Project Performance, brings more than 16 years of testing, quality assurance and process improvement experience. She is an American Society for Quality Certified Quality Manager and a Quality Assurance Institute Certified Quality Analyst. Focusing on process improvement and procedure development in the software testing and quality assurance areas, Ms. Chaney has successfully led process improvement, methodology development and re-engineering projects for organizations wishing to improve their software development, testing processes and tools implementation.

Thierry Ciot is a senior software engineer with more than 15 years of experience in software design and implementation. His experience spans many operating systems, programming languages (ADA, C, C++, Java, .NET) and a multitude of application domains, from mail and messaging to debugging tools. Mr. Ciot currently works at Compuware as a technical lead on DevPartner Studio. He is also the lead developer for System Compare, a tool that enables users to quickly find why an application works on one system but not on another.

Ross Collard is president of Collard & Co., a New York consulting firm. While he specializes in software testing and quality assurance, his consulting assignments have included strategic planning on the use of information technology for competitive advantage, the facilitation of quality improvement teams, management of large software development projects and the development of software engineering practices.

Mr. Collard has made keynote presentations at major software conferences, published articles and conducted seminars on information technology topics for businesses, governments and universities, including George Washington University, Harvard, New York University and U.C. Berkeley. He holds a B.E. in electrical engineering from the University of Auckland, New Zealand, an M.S. in computer science from the California Institute of Technology and an M.B.A. from Stanford.

Elfriede Dustin is an SQA manager at Symantec, author of the book "Effective Software Testing" and lead author of "Automated Software Testing" and "Quality Web Systems." She is currently writing the "Security Testing Handbook," along with two security experts, to be published by Symantec Press (spring 2006). She has also authored various white papers on the topic of software testing and is a frequent speaker at various software testing conferences.

Ms. Dustin has a B.S. in computer science and more than 15 years of IT experience in various positions, such as QA director for BNA Software and assistant director for integration test and deployment at CSC on the IRS modernization effort.

Michel Feaster is the director of product management for Mercury Interactive. She has seven years of experience as a systems engineer, and has worked closely with hundreds of Mercury Interactive software developers and QA manager customers helping them to streamline QA operations. She has spent a total of 12 years in the enterprise software industry. Ms. Feaster's in-depth technical knowledge and engaging style have made her a sought-after speaker at numerous industry events.

Jeff Feldstein is currently a manager of software development at Cisco Systems Inc. During his 24-year career, he has been a software developer, tester, development manager and computer consultant; for the past five years, he has been involved with software testing or has managed a team of developers who write software test tools. His specialties have included internetworking, real-time embedded systems, communications systems, hardware diagnostics and firmware, databases and test technologies. Mr. Feldstein has been one of the highest-rated speakers at previous Software Test & Performance conferences.

Matthew Fisher is a senior security engineer for SPI Dynamics and has been specializing in Web application security assessments for many years. A native Washingtonian, he has performed countless assessments of Web applications within the DoD and the federal government, as well as some of the largest commercial institutions around the globe, and is registered as a subject matter expert to the Defense Information Services Agency. Prior to joining SPI Dynamics, Mr. Fisher worked at Computer Sciences and Digex, where he acted as lead technical adviser on large-scale enterprise Web applications for Fortune 500 companies. He is a contributing author to the book "Google Hacking for Penetration Testers" and is currently working on his own book, titled "Web Application Security: A Guide for Developers and Penetration Testers." In addition, Mr. Fisher leads the Washington, D.C., OWASP chapter.

Aaron Flint has more than 10 years of increasingly senior quality assurance experience, primarily focused on testing enterprise-level server software, and responsible for functional testing, performance testing and building teams for quality assurance. For the past three years, Mr. Flint has been the QA manager at Layer7 Technologies. Prior to this, he worked at GTE Enterprise Solutions as QA team lead for multiple-listing-service real estate system software, and InfoWave Software as QA team lead and manager for enterprise wireless connectivity software.

Robert L. Galen is a senior QA manager at Thomson/Dialog in Cary, N.C.. He is also a principal of RGalen Consulting Group, and has held director-, manager- and contributor-level positions in both software development and quality assurance organizations. He has nearly 25 years of experience working in a wide variety of domains, from hard, real-time systems to Web-based information systems. Mr. Galen is an active member of ACM, ASQ, IEEE/CS and PMI and is a certified Scrum Master, a member of the Agile Alliance and the author of "Software Endgames" (Dorset House, 2005).

Andrew Glover is president of Stelligent, where his primary responsibilities include the strategic development of Stelligent's products and services. Mr. Glover's career includes founding

Vanward Technologies and leadership in software development for IBM, Philips Electronics and Procter & Gamble. He is a graduate of George Mason University in Fairfax, Va., and is a frequent speaker at industry events. Mr. Glover is a co-author of "Java Testing Patterns" (Wiley, 2004).

Robin F. Goldsmith has been president of the Go Pro Management consultancy since 1982. He works directly with and trains professionals in business engineering, requirements analysis, software acquisition, project management, quality assurance and testing. Previously he was a developer, a systems programmer/DBA/QA and a project leader with the City of Cleveland, leading financial institutions and a "Big 4" consulting firm. Author of numerous articles and the recent book "Discovering REAL Business Requirements for Software Project Success," Mr. Goldsmith was formerly international vice president of the Association for Systems Management and executive editor of the Journal of Systems Management. Mr. Goldsmith has an A.B. from Kenyon College, an M.S. from Pennsylvania State and a J.D. from Boston University.

Michael Hackett is a founding partner of LogiGear and is responsible for the direction and development of the company's training program. He has in-depth experience in software engineering and the testing of applications developed for deployment across multiple platforms. Mr. Hackett writes and teaches a software testing curriculum for LogiGear University, and for the U.C. Berkeley Extension. He is also co-author of "Testing Applications on the Web: Test Planning for Mobile and Internet-Based Systems," Second Ed., and holds a B.S. in engineering from Carnegie Mellon University.

Steven Haines started Quest Software's Java EE Performance Tuning professional services organization in 2002. Haines is the author of both "Java 2 Primer Plus" and "Java 2 From Scratch" and shares author credits on "Java Web Services Unleashed." He is also the author of two newly released books, "InformIT Java Reference Guide" and "Pro Java EE Performance Management." Haines has taught Java at the University of California in Irvine and at Learning Tree University. As the Java host on InformIT.com, Mr. Haines publishes a weekly column on everything from Java Web technologies to design patterns and performance tuning.

Linda Hayes is the CTO of Worksoft, and also the founder of three software companies, including AutoTester, which delivered the first PC-based test automation tool. Ms. Hayes holds degrees in accounting, tax and law, is an award-winning author on software quality, and has been a frequent industry speaker at numerous industry conferences and shows. She has been named one of Fortune Magazine's "People to Watch" and one of the "Top 40 Under 40" by Dallas Business Journal. She has been a columnist and contributor to StickyMinds and Better Software magazines, as well as a columnist for Computerworld and Datamation, author of the "Automated Testing Handbook" and co-editor of "Dare to Be Excellent" with Alka Jarvis on best practices in the software industry.

Joshua Hendrick recently joined Parasoft Professional Services team and has previously worked as a software engineer in Parasoft SOA Solutions group. He has contributed to the development of Parasoft Java-based SOA and Web services testing solutions, including development from an Eclipse environment. Mr. Hendrick earned his B.S. in computer science from the University of California, Davis, where he worked actively as a programmer in the Biological and Agricultural Engineering department research lab. His experience with SOA and Web services includes development of automated testing methodologies for SOA and working with numerous Parasoft customers worldwide to ensure secure, reliable and compliant Web services.

Chris Hetzler is employed in the Fargo, N.D., Microsoft Business Solutions development office as a software development engineer in test. On his latest project, Microsoft Dynamics GP Web Services, he was the lead technical tester and was responsible for the development of the product's test strategy. Mr. Hetzler has a B.S. in computer science and an M.S. in software engineering from North Dakota State University. He has won numerous public speaking awards throughout his academic career.

Ferhan Kilical is an experienced IT professional with more than 20 years of software development and engineering experience. She has successfully managed tuning efforts of many Web-based systems in government and nonprofit institutions. Early in her career, Ms. Kilical successfully managed high-profile projects for the Department of Defense. She has taught computer science courses in the Washington, D.C., metro area universities. Recently she started a Washington, D.C.-based consulting firm specializing in performance, load and stress testing, performance monitoring, application tuning and optimization.

Timothy Korson has had more than two decades of substantial experience working on a large variety of systems developed using modern software engineering techniques. This experience

includes distributed, real-time, embedded systems as well as business information systems in an *n*-tier, client/server environment. Dr. Korson's typical involvement on a project is as a senior management consultant with additional technical responsibilities to ensure high-quality, robust test and quality assurance processes and practices. Dr. Korson has authored numerous articles, and co-authored a book on Object Technology Centers.

Yury Makedonov was trained as a researcher and worked in an R&D organization dealing with composite materials. He has a Ph.D. degree in physics and mathematics. He is now using his skills and knowledge to improve software quality.

Dr. Makedonov has 10 years of testing experience. Currently, he is working as a QA manager, a test automation manager and a senior consultant for the Centre of Testing and Quality at CGI Group, a leading Canadian provider of end-to-end information technology and business process services.

Averil Meehan has a Ph.D. in the area of memory management of object-oriented programming languages. Currently she lectures at Letterkenny Institute of Technology, a third level college in Ireland with students at degree and masters levels. Dr. Meehan has presented conference papers and written articles for publications internationally, including "Performance Implications of Java Memory Management," "Computer Programming in a Blended Learning Environment," "Java Garbage Collection—A Generic Solution" and "The Semantics of Garbage Collection Rules, A Denotational Approach."

Hung Q. Nguyen is CEO and founder of LogiGear, a software quality engineering firm offering training, testing services and test automation products. He is author and co-author of several software testing books, including "Testing Applications on the Web," Second Ed., and "Testing Computer Software," Second Ed.

Mr. Nguyen writes and teaches a software testing curriculum for LogiGear University, as well as for U.C. Berkeley and the U.C. Santa Cruz Extension. He holds a B.S. in quality assurance from Cogswell Polytechnical College, is a graduate of a Stanford Graduate School of Business Executive Program, and is a certified quality engineer.

Thomas O'Mara has more than 25 years of experience with PC-based computing, ranging from Fiber Optic Gyroscope data acquisition using the stack-based FORTH language to Web-based applications utilizing the .NET Framework and ASP.NET. In between, there were C, Visual Basic and various database and middleware initiatives. Mr. O'Mara has been working with and writing articles about .NET technology since early 2001. He has considerable direct performance-tuning experience on a Web-based ASP.NET banking software application for large credit unions.

Ted Rivera is currently the global manager for quality assurance for the Automated Software Quality product suite for the Rational Products Division within IBM. He has been with IBM for more than 20 years as a developer, customer support manager and software testing manager. He currently holds two patents and has several others on file, and has contributed to the upcoming book "Agility and Discipline Made Easy: Best Practices From the Rational Unified Process," by Per Kroll and Bruce MacIsaac. Mr. Rivera is nearing completion of his Ph.D. from Southeastern Baptist Theological Seminary.

BJ Rollison is a technical trainer in the Engineering Excellence Group at Microsoft, where he designs and develops an intensive, hands-on technical training curriculum for new and experienced test engineers. He started his professional career in the industry working on developing custom solutions for hospitals and local government agencies in Japan. In 1994 he joined the Windows 95 project at Microsoft, focusing on the internationalization of the Windows operating system.

In 1996, Mr. Rollison became a test manager in the Internet Client and Consumer Division, responsible for several client products and a Web server. He moved to Microsoft's Internal Technical Training group in 1999 as the director of test training, responsible for planning and organizing training for more than 6,000 test engineers. He also teaches software testing courses at the University of Washington and sits on the advisory boards for software testing certificate programs at the University of Washington and Lake Washington Technical College.

Robert Sabourin, P.Eng., has more than 20 years of management experience leading teams of software development professionals to consistently deliver projects on time, on quality and on budget.

Mr. Sabourin is an adjunct professor of software engineering at McGill University who often speaks at conferences around the world on software engineering, SQA, testing and management issues.

Neeraj Sangal is president of Lattix, a company specializing in software architecture management solutions and services.

Register at www.stpcon.com

He has analyzed many large proprietary and open-source systems. Prior to Lattix, Mr. Sangal was president of Tendril Software, which pioneered model-driven EJB development and synchronized UML models for the Java programming language. Tendril was acquired by BEA/WebGain. Prior to Tendril, he managed a distributed development organization at Hewlett-Packard.

Paul Slauenwhite is a software developer on the IBM Autonomic Computing (AC) Tools and Technologies project and a committer to the Eclipse Test and Performance Tools Platform (TPTP) open-source project. After receiving a B.Sc. in computer science from Dalhousie University, Mr. Slauenwhite joined IBM in 2000 and worked on the WebSphere Object Level Trace (OLT) project. In 2001, he joined the IBM WebSphere Studio Team and developed logging and tracing technologies. He is currently working on an M.Math in software engineering at the University of Waterloo.

Over the past 15 years, ***Peter Sody*** has been a testing contractor, software developer, test engineer and development manager for several companies and organizations in Germany and the United States, most recently for Eclipsys and Vertex. Being an advocate of test-driven development, he is now using his skills for developing architectures and frameworks for both functional test automation and performance testing. Mr. Sody has authored publications on various aspects of software development, including presentations at conferences such as IEEE RE and Net.ObjectDays. He holds an M.S. in computer science from Kaiserslautern University, Germany.

Now an independent software consultant, ***Alfred Sorkowitz*** was a computer scientist with the Department of the Navy, responsible for developing real-time, software-intensive embedded systems. Prior to joining the Department of the Navy, he was director of the Standards and Quality Control Staff, U. S. Department of Housing and Urban Development. Mr. Sorkowitz has published papers and has presented seminars on software metrics, SQA and testing at conferences sponsored by the IEEE Computer Society, ACM and the British Computer Society.

Dean Stevens has been involved with developing and delivering world-class hardware, software and service products for more than 20 years. Prior to joining Symbio, Mr. Stevens

founded and served as the CEO of China TechSource—an outsourcing broker for Chinese services firms. In addition, he has operated a consulting firm that worked with corporations to resolve executive management and execution issues. He has demonstrated success managing global remote development, multicompany projects and distributed virtual organizations. Mr. Stevens began his career writing FORTRAN code for a CDC mainframe. He is a graduate of the University of Idaho.

Tom Stracener was one of the founding members of nCircle Network Security. While at nCircle he served as the head of vulnerability research from 1999 to 2001, developing one of the industry's first quantitative vulnerability scoring systems, and co-inventing several patented technologies. Mr. Stracener is an experienced security consultant, penetration tester and vulnerability researcher. One of his patents, "Interoperability of vulnerability and intrusion detection systems," was granted by the USPTO in October 2005. He is the senior security analyst for Cenzic's CIA Labs and the architect of Cenzic's Application Penetration Testing Methodology.

Mary Sweeney has been developing, using and testing relational database systems for more than 20 years for such companies as Boeing and Software Test Labs. She's the author of "Visual Basic for Testers" (Apress, 2001) and "A Tester's Guide to .NET Programming" (Apress, 2006). Ms. Sweeney is a college professor with a degree in mathematics and computer science from Seattle University. She is an MCP in SQL Server, and is on the board of IIST (International Institute of Software Test).

Stanley Au Yeung has years of IT experience in database applications, performance testing and quality assurance. After getting his M.S. degree from George Washington University in Information Technology, he was heavily involved with research and application development projects in the university. Then he joined a nonprofit organization where he mastered skills and knowledge in quality assurance and performance testing. He was involved with various high-profile Web systems tuning and optimization projects. He is currently working on a performance optimization project for a mission-critical Navy application as a senior consultant.

Scott Will is the manager of the Quality Engineering Team for the Tivoli Software Products Division within IBM. He has been with IBM for more than 15 years and was previously an Air Force combat pilot. He has held positions as chief programmer, customer support team lead and software test manager. He holds three patents and has several others on file, and has also contributed to the upcoming book "Agility and Discipline Made Easy: Best Practices From the Rational Unified Process," by Per Kroll and Bruce MacIsaac. Mr. Will graduated from Purdue University with degrees in computer science, mathematics and numerical analysis. He also holds an M.B.A. from Wayland Baptist University.

Hotel

The Software Test & Performance Conference has secured a special rate of $159/single per night with the Hyatt Regency Cambridge. To make your reservations, use the link on the show's Web site, www.stpcon.com. You can also call the hotel directly. Be sure to identify yourself as being with the Software Test & Performance Conference to receive the group rate.

YOU MUST MAKE YOUR RESERVATIONS BY OCTOBER 16 TO SECURE THIS RATE.

Hyatt Regency Cambridge
575 Memorial Drive
Cambridge, MA 02139-4896
Tel: +1-617-492-1234
Fax: +1-617-491-6906

Directions

From Logan International Airport:

Follow signs to Boston via Sumner Tunnel. After you come out of the tunnel take exit 26A—(Storrow Drive/Back Bay/Cambridge). Keep left, go onto Storrow Drive. Go 3/4 mile and take exit on left-hand side (Government Center/Kendall Square). At the top of the ramp take a right over the Longfellow Bridge; at the end of the bridge turn right onto Memorial Drive West (Rt. 3 North). Stay on Memorial Drive approximately 1 1/2 miles; at the third traffic light turn right. The entrance to the Hyatt will be on your left.

Transportation

From the Boston Airport:

- Taxi – Approximately $35
- Subway – $1.25 (T stops), 1 1/2 miles from hotel (taxi available)
  Above Ground: B.U. Central, Green Line (T stops)
  Underground: Harvard Square, Kendall Square Red Line

Amtrak:

- Back Bay Station is located 3 miles from hotel, approximately 10-15 minutes away. Amtrak also stops at South Station in the Financial District, which is approximately 5 miles, 20-30 minutes, from hotel. Taxi is available to and from the hotel. From Back Bay Station, cost is $7-$10. From South Station, cost is $12-$15.

Downtown Cambridge:

- Hotel offers complimentary scheduled shuttle service to locations in Cambridge. The shuttle service will take guests anywhere from Harvard Square to the Cambridgeside Galleria Mall area and all points between.
- For guests who would like to get into the city of Boston, the shuttle will stop at Kendall Square T Stop (Red Line) or the Boston University T Stop (Green Line). For more information or to sign up for the shuttle, contact the Guest Services Desk at in-house extension 51 or call +1-617-492-1234.

General Shuttle Information:

Traffic conditions may change pick-up times, forcing the shuttle off schedule, especially during rush hour and inclement weather. The Hyatt shuttle is for registered guests only. If requested, please show your room key to the driver as identification. T Stop schedules are available at the Guest Services Desk. For the safety of our passengers, luggage is not allowed on the shuttle. Shuttle schedule is subject to change without notice.

MBTA: (Subway) 1 1/2 miles from hotel.

Parking:

Self-parking is $28 per night with in/out privileges (clearance 6' 8").
Valet parking is $30 per night; RV and van parking is available at your own risk in the employee parking lot, based on availability. See the doorman or front desk upon arrival for details.

Register at www.stpcon.com

MONDAY November 6

REGISTRATION OPEN
4:00 pm - 7:00 pm

TUESDAY November 7

REGISTRATION OPEN
7:30 am - 7:00 pm

TUTORIALS
9:00 am - 5:00 pm

T-1	Assessing Your Test Team Effectiveness, Efficiency and More - Black
T-2	Testing Techniques: Theory and Application - Rollison
T-3	Testing in Highly Iterative, Quasi-Agile Projects—Practical Strategies for Mixed Culture Projects - Korson
T-4	Twenty-One Ways to Spot—And Fix—Requirements Errors Early - Goldsmith
T-5	Foundations of GUI Test Automation - Makedonov
T-6	Software Endgames: How to Finish What You've Started - Galen
T-7	Using Metrics to Improve Software Testing - Sorkowitz
T-8	Creating Agility and Effectiveness in Software Testing - Hayes

BIRDS OF A FEATHER SESSION
8:00 pm - 10:00 pm

WEDNESDAY November 8

REGISTRATION OPEN 7:30 am - 7:00 pm

INDUSTRY KEYNOTE 4:30 pm - 4:45 pm — KEYNOTE: REX BLACK
EXHIBITS OPEN 2:30 pm - 7:00 pm — ATTENDEE RECEPTION

TECHNICAL CLASSES

8:30 am - 10:00 am		10:30 am - 12:00 pm		1:15 pm - 2:30 pm	
101	Seven Low-Overhead Software Process Improvement Methods - Goldsmith	201	Prevent Showstopper Overruns With Risk-Based Proactive Testing - Goldsmith	301	Hacking 101: Donning the Black Hat to Best Protect Applications From Today's Hacking Threats - Stracener
102	Just-in-Time Testing Techniques and Tactics, Part 1 - Sabourin	202	Just-in-Time Testing Techniques and Tactics, Part 2 - Sabourin	302	Five Core Metrics to Guide Your Software Endgames - Galen
103	Automated Database Testing: Testing and Using Stored Procedures - Sweeney	203	How to Turn Your Testing Team Into a High-Performance Organization - Hackett	303	Overcoming Requirements-Based Testing's Hidden Pitfalls - Goldsmith
104	Lessons Learned in Test Automation, Part 1 - Dustin	204	Lessons Learned in Test Automation, Part 2 - Dustin	304	Managing Acceptance Testing Cycles More Efficiently - Makedonov
105	Code Coverage Metrics and How to Use Them - Black	205	Database Security: How Vulnerable Is Your Data? - Sweeney	305	How to Optimize Your Web Testing Strategy - Nguyen
106	Foundations of GUI Test Automation Using C#, Part 1 - Rollison	206	Foundations of GUI Test Automation Using C#, Part 2 - Rollison	306	Model-Based Testing for Java and Web-based GUI Applications - Feldstein
107	Designing for Testability - Feldstein	207	Analyze the Return on Your Testing Investment - Black	307	Taking AIM—Using Visual Models for Test Case Design - Sabourin
108	Testing the Software Architecture - Sangal	208	Using Scrum to Manage the Testing Effort - Galen	308	Elements of Software Design for Unit Testing - Ciot
109	Effectively Training Your Offshore Test Team - Hackett	209	Quality Throughout the Software Life Cycle - Feldstein		

BIRDS OF A FEATHER SESSION 8:30 pm - 10:30 pm

Software Test & Performance CONFERENCE

Last Year's Conference SOLD OUT! Register Early!

Register at www.stpcon.com

THURSDAY
November 9

REGISTRATION OPEN 7:30 am - 4:00 pm

EXHIBITS OPEN 12:00 pm - 4:00 pm

4:45 pm - 5:30 pm

5:30 pm - 7:00 pm

TECHNICAL CLASSES

3:00 pm - 4:15 pm		8:30 am - 10:00 am		10:30 am - 12:00 pm		2:00 pm - 3:15 pm		3:45 pm - 5:00 pm	
401	The Secure Software Development Life Cycle - Dustin	501	Strategies and Tactics for Global Test Automation, Part 1 - Nguyen	601	Strategies and Tactics for Global Test Automation, Part 2 - Nguyen	701	The Five Most Dangerous Application Security Vulnerabilities—and How to Test for Them - Basirico	801	Effective Metrics for Managing a Test Effort - Bradshaw
402	Accelerate Testing Cycles With Collaborative Performance Testing - Cavallaro	502	Creating and Leading the High-Performance Test Organization, Part 1 - Galen	602	Creating and Leading the High-Performance Test Organization, Part 2 - Galen	702	S-Curves and the Zero Bug Bounce: Plotting Your Way to More Effective Test Management - Bradshaw	802	Software Testing a Service-Oriented Architecture - Rivera, Will
403	Deciding What Not to Test - Sabourin	503	Pinpointing and Exploiting Specific Performance Bottlenecks - Barber	603	SOA Performance Testing Challenges - Barber	703	Testing Java Programs—Memory Management Issues - Meehan	803	Real-World Performance Testing Lab for (Almost) Free - Flint
404	Putting the User Back in User Acceptance Testing - Goldsmith	504	Performance Tuning ASP.NET 2.0 Applications, Part 1 - O'Mara	604	Performance Tuning ASP.NET 2.0 Applications, Part 2 - O'Mara	704	Defining Test Data and Data-Centric Application Testing - Hetzler	804	Testing Java Applications Using the Eclipse Test and Performance Tools Platform (TPTP) - Slauenwhite
405	Getting a Handle on Risk: Risk-Based Testing Strategies - Chaney	505	Identify and Mitigate Risks Through Testing, Part 1 - Black	605	Identify and Mitigate Risks Through Testing, Part 2 - Black	705	Coding Standards and Unit Testing—Why Bother? - Hendrick	805	Building a Bridge Between Functional Test Automation and Performance Testing - Sody
406	Java EE Performance Tuning Methodology: Wait-Based Tuning - Haines	506	Rapid Business-Driven Testing - Chaney	606	Exploiting Web Application Code: The Methodologies and Automation of SQL Injection - Fisher	706	The 5% Challenges of Test Automation - Buwalda	806	Best Practices for Managing Distributed Testing Teams - Stevens
407	Agile Test Development - Buwalda	507	Unit Testing for Agile Development, Part 1 - Sabourin	607	Unit Testing for Agile Development, Part 2 - Sabourin	707	Performance Testing for Managers - Barber	807	Diagnosing and Resolving J2EE Application Issues Before Deployment - Kilical, Au Yeung
408	Techniques for Testing Packaged Application Performance - Feaster	508	Using Code Metrics for Targeted Code Refactoring - Glover	608	Verifying Software Robustness - Collard	708	What Hollywood Can Teach You About Software Testing - Sabourin		
		509	Recruiting, Hiring, Motivating and Retaining Top Testing Talent - Feldstein	609	Metrics: How to Track Things That Matter - Chaney				

Pricing and Registration

Register Early and SAVE!

	eXtreme Early Bird	Super Early Bird	Early Bird	Full Price
Register By	By July 28	By Sept. 15	By Oct. 20	After Oct. 20
Full Event Passport November 7-9	$1,095 Best Value	$1,195	$1,365	$1,565
Two-Day Technical Conference Only November 8-9	$935	$1,035	$1,195	$1,345
Tutorials Only November 7	$695	$795	$845	$945
Exhibits Only November 8-9	FREE	FREE	FREE	$50

All prices are in US dollars

How to Register

REGISTER ONLINE. Register online at www.stpcon.com and use one of the following payment methods:

CREDIT CARD. You can use the secure online form to pay via credit card and get immediate confirmation of your classes. MasterCard, Visa and American Express are accepted cards. You'll receive a REGISTRATION RECORD and RECEIPT. Please print out these pages and bring them with you to the conference. Present them at the Registration Desk to pick up your badge and any course materials.

CHECK. Fill out the online registration form. Print out the REGISTRATION RECORD and RECEIPT and mail to BZ Media LLC, 7 High Street, Suite 407, Huntington, NY 11743. Be sure to include your payment. Online registrations that are mailed without payment will not be confirmed until payment is received.

P.O. If you register using a P.O., you will be invoiced immediately for the registration amount. Payment must be received before your registration can be confirmed.

GROUP DISCOUNTS. Registering four or more attendees from your company? You can receive a $100 discount off the Full Event Passport on each registration. Enter the word GROUP when asked for a code on our online registration form.

ALUMNI DISCOUNT. Have you attended the Software Test & Performance Conference in previous years? Welcome back! Enter the code ALUMNI to receive $100 off the Full Event Passport registration price.

REFUND POLICY. You can receive a full refund, less a $50 registration fee, for cancellations made by October 10, 2006. Cancellations after this date are non-refundable. Send your cancellation in writing to registration@bzmedia.com.

Paid Conference/Tutorial Registration Includes:

- Admission to tutorials and/or technical classes. Please make your class selections when registering.
- Admission to keynotes
- Admission to exhibits
- Conference materials
- Attendee reception
- Continental breakfast, coffee breaks

Exhibits-Only Registration Includes:

- Admission to keynotes
- Admission to exhibits
- Attendee reception

Register Online Today at www.stpcon.com!

Registration Questions

Contact Donna Esposito at +1-415-785-3419 or desposito@bzmedia.com.

Software Test & Performance CONFERENCE

BZ Media LLC
7 High Street
Suite 407
Huntington, NY 11743