

July/August 2008

\$9.95 [www.StickyMinds.com](http://www.StickyMinds.com)

# BETTER SOFTWARE

The Print Companion to [StickyMinds.com](http://StickyMinds.com)

**HIP-HIP HOORAY!**  
Praise for ambiguity

**DOUBLE YOUR FUN**  
8 Benefits of  
pair programming



**HOW TO**  
**FAIL**  
**WITH AGILE**  
**20** Tips to Help You  
Avoid Success

**FIND THE BUG INSIDE & WIN  
AN IPOD SHUFFLE™!**

# SERIOUS SOURCECODE ANALYSIS

Klocwork helps software developers create better code.

When developing mission-critical embedded software, developers must quickly and accurately identify, assess and fix critical software bugs and security vulnerabilities right at their desktop before they impact anyone else.

Klocwork's leading static source code analysis tools provide powerful, collaborative analysis of your C, C++ and Java code at the earliest point in the software development process – before code check-in – when detected issues are easiest and less costly to fix, freeing developers to do what they do best – innovate.

Take the first step towards better code. Visit [www.klocwork.com/freetrialsignup](http://www.klocwork.com/freetrialsignup) for a free product trial.

**Klocwork**

[www.klocwork.com](http://www.klocwork.com)







**Take quality to the next level**



## **DevTest Studio**

The integrated solution for defect tracking, test management and automated testing

### **DevTrack**

Use DevTrack to track defects/issues

- Track each issue through a definable workflow
- SCM integration – track fixes against their source code deliverables
- Deploy a resolution across multiple releases, versions and products
- Reporting and metrics to illustrate the entire defect lifecycle

### **DevTest**

Use DevTest to manage your testing

- Create a central repository for your test cases, knowledge items and automation scripts
- Schedule releases and test cycles using a wizard-driven interface
- Execute test assignments and submit defects from the same interface
- Track results with real-time dashboards and reports

### **TestLink**

Use TestLink to automate your testing

- Add automated tests to the DevTest test library
- Schedule automated tests along with manual tests
- Launch automated tests from the DevTest interface
- Track automation results with real-time dashboards and reports

**TechExcel**

[www.techexcel.com](http://www.techexcel.com) | 1-800-439-7782



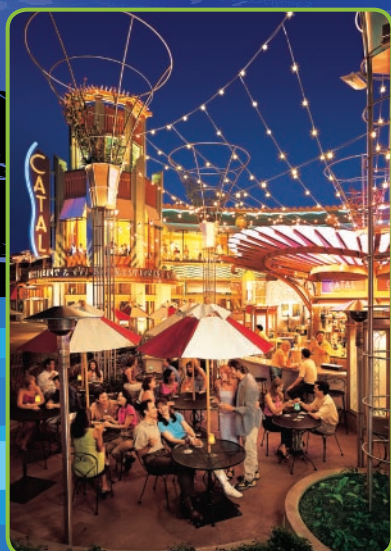
**STAR  
WEST**

# SOFTWARE TESTING

ANALYSIS & REVIEW

*The Greatest Software  
Testing Conference on Earth*

Anaheim,  
**California**



[www.sqe.com/STARWEST](http://www.sqe.com/STARWEST)

**Register Early and save \$200!**

**September 29–October 3, 2008**

**Disneyland® Hotel**

**98% of 2007 Attendees  
Recommend STARWEST  
to Others in the Industry**



As to Disney photos, logos, properties: ©Disney

## KEYNOTES



**Testing Lessons from Springfield—Home of the Simpsons**

*Rob Sabourin, AmiBug.com*



**Telling Your Exploratory Story**

*Jon Bach, Quardev, Inc.*



**Six Thinking Hats for Software Testers**

*Julian Harty, Google*



**Branch Out Using Classification Trees**

*Julie Gardiner, Grove Consultants*



**Has the Time for the Adversarial Organization Passed?**

*Gerard Meszaros, ClearStream Consulting*



**Testing Microsoft Office: Experiences You Can Leverage to Drive Quality Upstream**

*Tara Roth, Microsoft*





Find the Bug  
and win an iPod Shuffle™!  
Search through the digital  
edition to find the flying  
bug. Click it to be entered for a  
chance to win an iPod Shuffle™.

## Cover Story

### HOW TO FAIL WITH AGILE 24

A switch to agile often conflicts with personal career goals such as maintaining the status quo and working no harder than necessary. These twenty guidelines will help you sabotage your agile project, helping you fail quickly and spectacularly.

by Clinton Keith and Mike Cohn

## Features



### A GALAXY OF PATTERNS 30

The Gang of Four's design patterns have a special place in many programmers' hearts. But it's time to look beyond the GoF twenty-three and realize they aren't the only patterns in the universe. *by Neil Harrison*

### GIVE YOUR DEFECTS SOME STATIC—USING AUTOMATED STATIC ANALYZERS TO DEBUG YOUR CODE 36

Computer security has raised the demand for automated tools that can analyze source code for vulnerabilities and defects. Find out how you can put automated static analyzers to work for you. *by Greg Pope, Kim Ferrari, and Bill Oliver*

## Columns & Departments

### In Every Issue

Mark Your Calendar 4

Contributors 6

eLightenment 12

Product Announcements 43

10 Things You Might  
Not Know About ... 46

Ad Index 48

**Better Software magazine**—The print companion to StickyMinds.com brings you the hands-on, knowledge-building information you need to run smarter projects and deliver better products that win in the marketplace and positively affect the bottom line.

**Subscribe today to get ten issues.**

Visit [www.BetterSoftware.com](http://www.BetterSoftware.com)  
or call 800.450.7854.

### TECHNICALLY SPEAKING 9

**Software: Use at Your Own Risk** • *by Chuck Allison*

When was the last time you read a software license agreement? We need to move away from "Use at your own risk" software and be upfront with customers about the true cost of quality.

### CODE CRAFT 16

**Programming with GUTs** • *by Kevlin Henney*

A Good Unit Test needs both to illustrate and define the behavioral contract of the unit in question. Do you have GUTs?

### TEST CONNECTION 20

**Two Cheers for Ambiguity** • *by Michael Bolton*

One tester's ambiguity is another tester's gauge for assessing consensus on a project and how to achieve that consensus.

### MANAGEMENT CHRONICLES 22

**Going on a Picnic with James Watt** • *by Clarke Ching*

Find out what picnic planning and steam engines have to do with delighted customers.

### THE LAST WORD 47

**Encourage Pair Programming** • *by Rob Myers*

Thinking about trying pair programming? Here are several reasons why you should.

**StickyMinds.com** We invite you to visit StickyMinds.com, the online companion to *Better Software* magazine.

StickyMinds.com covers the same pertinent topics as the magazine, putting the power of information at the click of your mouse. Weekly columns, headline-making bugs, hundreds of technical papers, an online tools guide, discussion boards, and so much more make StickyMinds.com your site for 24/7 brainfood to help you build better software.

## MARK YOUR CALENDAR

### TRAINING WEEKS

[www.sqetraining.com/Public](http://www.sqetraining.com/Public)

#### Testing

**September 8–12, 2008**

New York, NY

**September 15–19, 2008**

Washington, DC

#### Agile Software Development

**September 8–12, 2008**

Washington, DC

**September 22–26, 2008**

Denver, CO

### SOFTWARE TESTING CERTIFICATION

[www.sqetraining.com/certification](http://www.sqetraining.com/certification)

**August 26–28, 2008**

Boston, MA

**September 8–10, 2008**

New York, NY

**September 9–11, 2008**

Minneapolis, MN and Salt Lake City, UT

**September 23–25, 2008**

Philadelphia, PA and Atlanta, GA

### CONFERENCES

#### STARWEST 2008

##### Software Testing Analysis & Review

[www.sqe.com/starwest](http://www.sqe.com/starwest)

**September 29–October 3, 2008**

Disneyland Hotel

Anaheim, CA

#### Agile Development Practices 2008

[www.sqe.com/agiledevpractices](http://www.sqe.com/agiledevpractices)

**November 10–14, 2008**

Shingle Creek Resort

Orlando, FL

# BETTER SOFTWARE

Publisher

**Wayne Middleton**

Vice President of Publishing

**Holly N. Bourquin**

#### Editorial

Editor

**Heather Shanholtzer**

Managing Technical Editor

**Lee Copeland**

Technical Editors

**Chuck Allison**

**Jonathan Kohl**

**Antony Marciano**

Editor, StickyMinds.com

**Francesca Matteu**

Managing Editor, Multimedia

**Joseph McAllister**

Production Coordinator

**Cheryl M. Burke**

#### Design

Creative Director

**Catherine J. Clinger**

#### Advertising

Senior Advertising Sales Manager

**Shae Young**

Advertising Sales Manager

**Joe Anderson**

Production Coordinator

**April Evans**

#### Circulation and Marketing

Marketing Coordinator

**Megan Brown**

Circulation Coordinator

**Jamie Green-Gago**

A PUBLICATION OF SOFTWARE QUALITY ENGINEERING



#### CONTACT US

Editors: [editors@betersoftware.com](mailto:editors@betersoftware.com)

Subscriber Services: [info@betersoftware.com](mailto:info@betersoftware.com)

Phone: 904.278.0524, 888.268.8770

Fax: 904.278.4380

Address:

Better Software magazine  
Software Quality Engineering, Inc.  
330 Corporate Way, Suite 300  
Orange Park, FL 32073



# To load test your website, you could type this:

## Definitions

! Standard Defines

```
Include "RESPONSE_CODES.INC" Include "GLOBAL_VARIABLES.INC" CHARACTER*512 USER_AGENT Integer  
USE_PAGE_TIMERS CHARACTER*256 MESSAGE Timer T_OBFUSCATED CHARACTER*1024 cookie_2_0 CHARACTER*1024  
cookie_2_1 CHARACTER*1024 cookie_15_0 CONSTANT DEFAULT_HEADERS = "Host: " + domain + ".com^J" &  
"Accept-Encoding: gzip, deflate^J" & "User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;  
.NET CLR 1.1.4322; .NET CLR 2.0.50727)" CONSTANT S_cookie_1_0 = "B=1p1c30p38in6q&b=3&s=7m" CONSTANT  
S_cookie_1_1 = "YLS=v=1&p=0&n=1" CONSTANT S_cookie_1_2 = & "F=a=60rfgt4MvT9diVoFZzhX7Wriqn1T2n2489Sx"  
CONSTANT S_cookie_1_3 = "PH=fn=fnP28mrdGyGdwWauu9qREw--&l=en-US" CONSTANT S_cookie_1_4  
CONSTANT S_cookie_1_5 = "U=mt=dsuqeJ2MhYrp3Y7EjKt4qCUZiUWDW.w4_.E&ux=JHnhHB&un=24a238gchl7lv"
```

## Code

```
!Read in the default browser user agent field  
Entry[USER_AGENT,USE_PAGE_TIMERS] Start Timer T_OBFUSCATED PRIMARY GET URI "http://" +  
".com/ HTTP/1.1" ON 1 & HEADER DEFAULT_HEADERS & WITH {"Accept: image/gif, image/x-bitmap,  
image/jpeg, image/pjpeg, " & "application/x-shockwave-flash, application/msword, /*", &  
"Accept-Language: en-us", & "Connection: Keep-Alive", & "Cookie: "+S_cookie_1_0+";  
"+S_cookie_1_1+"; "+S_cookie_1_2+"; "+S_cookie_1_3+";" DISCONNECT FROM 1 WAIT 391
```

or this:

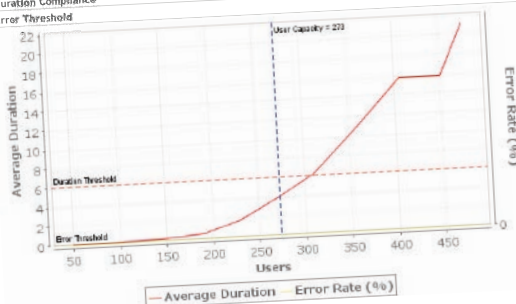
[www.webperformanceinc.com](http://www.webperformanceinc.com)

## Test Report: 9/12/06 1:52 PM Acceptance Test

### User Capacity

The User Capacity section shows the measured user capacity, or how many users the web-based application could take while still meeting performance requirements. How this measurement was arrived at is demonstrated in the chart, which shows the duration and error thresholds. The place at which the measured duration and error count intersect the thresholds is the measured capacity.

Estimated User Capacity 273  
Settings 6 sec  
Duration Threshold 95%  
Duration Compliance 0%  
Error Threshold



Users	Thresholds Satisfied	Error Rate	Maximum Duration (ms)	Duration Compliance
50	yes	0.00%	00:00.148	100.00%
95	yes	0.00%	00:00.140	100.00%
143	yes	0.00%	00:00.174	100.00%
190	yes	0.00%	00:00.589	100.00%
229	yes	0.00%	00:01.721	100.00%



## WEB PERFORMANCE LOAD TESTING

Why code every test case by hand, when our unique software detects and automatically configures the test cases for you – quickly and accurately, then gives you superior reports that are easy to understand? With Web Performance automatic load testing, the time and money you save could increase productivity as much as 500 percent.

For more information about how you can increase performance and productivity using Web Performance automated load testing, visit [www.webperformanceinc.com](http://www.webperformanceinc.com)



**CHUCK ALLISON** developed software for more than twenty years before becoming a professor of computer science at Utah Valley University. He is a technical editor for *Better Software* magazine and founding and current editor of the online journal *The C++ Source*. He spent most of the 1990s as an active member of the C++ Standards Committee and is author of two C++ books, including *Thinking In C++, Volume 2: Practical Programming*, with Bruce Eckel. His company, Fresh Sources, Inc., gives onsite training in C++, Python, and design patterns. His current top technical interest is the resurgence of functional programming. Whenever he finds a little down time he plays classical guitar or bikes the country roads of central Utah. Contact Chuck at [chuck@freshsources.com](mailto:chuck@freshsources.com).



**MICHAEL BOLTON** lives in Toronto and teaches heuristics and exploratory testing in Canada, the United States, and other countries. He is co-author, with James Bach, of *Rapid Software Testing* and a regular contributor to *Better Software* magazine. Contact Michael at [mb@developsense.com](mailto:mb@developsense.com).



**ROB MYERS** has more than twenty years of professional experience in software development. Rob is an Extreme Programming coach who brings to the classroom his passion for value-oriented software development, team development, and sane work environments. He teaches *Test-Driven Development*, *Design Patterns Explained*, *Implementing Scrum for Your Team*, *Lean-Agile Testing*, and other courses.



**CLARKE CHING** is a New Zealander living in Scotland. In addition to being an independent consultant and a regular columnist on *StickyMinds.com*, he's a passionate advocate of agile software development and a chairman of the AgileScotland special interest group, which meets monthly in Edinburgh. Clarke currently is writing a book titled *Rolling Rocks Downhill*, in which he explains why working with software projects often feels like pushing rocks uphill. He also demonstrates how to use lean, quality, and agile techniques to make your projects more productive and predictable. Read more about his book and other articles and listen to his podcasts at [www.clarkeching.com](http://www.clarkeching.com).



**MIKE COHN** is the founder of Mountain Goat Software, a process and project management consultancy that specializes in helping companies adopt and improve the use of agile processes and techniques. He is the author of *Agile Estimating and Planning* and *User Stories Applied for Agile Software Development*. Mike is a founding member of the Agile Alliance, serves on its board of directors, and is a frequent presenter at the STAR and Better Software conferences. He can be reached at [mike@mountaingoatsoftware.com](mailto:mike@mountaingoatsoftware.com).



**KIMBERLY FERRARI** has more than twenty years of software development experience in both the commercial and government sectors. Kim has held positions from programmer to software engineering lead for a team of twelve developers. Currently Kim works for the University of California at the Lawrence Livermore National Laboratory where she is the security and protection group leader and software engineering lead for the security and protection group. Prior to this position, she worked for Lockheed Martin. Kim programs in many languages including C#, C++, and Ada.





NEIL HARRISON is an assistant professor of computer science at Utah Valley University in Orem, Utah. Before that, he developed software professionally for more than twenty years. Neil is the co-author of *Organizational Patterns of Agile Software Development*. He is acknowledged as the world's leading expert on pattern shepherding, and the Pattern Languages of Programs (PLoP) Conference shepherding award is named after him. Email Neil at [harrisne@uvsc.edu](mailto:harrisne@uvsc.edu).



KEVLIN HENNEY is an independent consultant and trainer based in the UK. He provides consultancy and training in programming techniques, software architecture, and development process. He is co-author of two recent books on patterns, *A Pattern Language for Distributed Computing* and *On Patterns and Pattern Languages*.



Over the course of twenty years, CLINTON KEITH has gone from programming avionics for advanced fighter jets and underwater robots to overseeing programming for hit video game titles such as "Midtown Madness," "Midnight Club," and "The Bourne Conspiracy." He introduced the video game industry to agile development and is now an agile coach teaching teams how to adopt Scrum. Clinton's Web site is [www.ClintonKeith.com](http://www.ClintonKeith.com).



WILLIAM OLIVER received a BS in mathematics from the University of Utah in 1974. He served as a nuclear submarine officer until 1979 and worked in the aerospace industry as a test engineer on solid fuel rocket engines for three years. William worked in the process control industry for approximately fifteen years before joining Lawrence Livermore National Laboratory in February of 1996. Since then he has completed all course work for a master's degree in computer science and has developed requirement specifications for control systems, been a software developer for scientific applications that run on the world's largest supercomputers, and is currently a software quality engineer supporting software quality assurance practices for scientific codes. He is the project administrator for a static analysis tool and assists various code teams in performing software defect analysis.



GREGORY POPE has more than thirty-five years of experience applying common sense to developing software in the commercial and government sectors. Greg has held positions from programmer to CEO. He has consulted, presented, and taught software quality worldwide and also holds patents for automated software testing systems. Currently Greg works for the Lawrence Livermore National Laboratory where he is software quality engineering group leader and V&V project leader for advanced simulation and computing (ASC).



**BETTER  
SOFTWARE**

# Agile Development Practices

[www.sqe.com/agiledevpractices](http://www.sqe.com/agiledevpractices)

**November 10-14, 2008  
Orlando, FL  
Shingle Creek Resort**

Meet agile experts first hand as they share their insight and wisdom about the best practices and techniques in the agile industry.

For additional information visit:

[www.sqe.com/agiledevpractices](http://www.sqe.com/agiledevpractices)

**Great Speakers.  
Great Content.  
Great Location.**

As agile software development moves beyond the early-adopter stage and becomes accepted and used throughout the industry, we at Software Quality Engineering want to be a part of this revolution. Whether you're investigating agile concepts or committed to agile practices in your organization, this conference has valuable information for you.

## Covering These Hot Agile Topics

Agile Methods and Processes  
Transitioning to Agile  
Agile Leadership Principles  
Managing Agile Projects with SCRUM  
Estimating in Agile Projects  
Lean Software Development  
Organizational Models for Agile  
Agile Release Planning  
User Stories and Use Cases  
Agile Design Approaches  
Pair Programming  
Agile Teams  
Measuring Agility  
Continuous Integration  
Test-Driven Development  
Tester Roles in Agile  
Distributed Agile Development  
Collaboration Methods  
Personal Skills Development  
Behavior-Driven Development  
Becoming a Change Agent  
Scaling Agile Projects  
Hiring and Developing Agile Teams  
And much more ...

**To Register Call 888.268.8770  
or 904.278.0524 or Visit  
[www.sqe.com/agiledevpractices](http://www.sqe.com/agiledevpractices)**



# Software: Use at Your Own Risk

by Chuck Allison

“BANGKOK (Reuters)—Security guards smashed their way into an official limousine with sledgehammers on Monday to rescue Thailand’s finance minister after his car’s computer failed ... All doors and windows had locked automatically when the computer crashed, and the air-conditioning stopped, officials said. ‘We could hardly breathe for over 10 minutes ... It took my guard a long time to realize that we really wanted the window smashed so that we could crawl out. It was a harrowing experience.’” [1]

Many of us merely reboot when software misbehaves, but, as one CNN report stated, “malfunctions caused by bizarre and frustrating glitches are becoming harder and harder to escape now that software controls everything from stoves to cell phones, trains, cars, and power plants.” [2]

For years, users of software products have blithely glanced over license agreements such as the following: “[The Company] shall not be liable in any manner whatsoever for results obtained for using this software ... THESE MATERIALS ARE PROVIDED ‘AS IS’ WITHOUT WARRANTY OF ANY KIND ... [THE COMPANY] AND ITS SUPPLIERS DISCLAIM ANY AND ALL WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED.”

Contrast this with the warranty for almost any other product you buy. When purchasing an appliance, the strength of the warranty is often what sways the buyer’s decision. With software, it’s always “use at your own risk.” Is it any wonder we’re conditioned to expect software to fail?

Is it really all that hard to produce software that works? Yes. Software development has been likened to nailing jelly to a tree—and for good reason. Is

“Software development has been likened to nailing jelly to a tree—and for good reason.”

that an excuse for poor software quality? Not from where I sit. I think we are approaching a time when one-sided license agreements will no longer fly.

The same CNN article reports that “defects stem from several sources: software complexity, commercial pressure to bring products out quickly, the industry’s lack of liability for defects, and poor work methods.” The complexity stems from the increased applicability of automation to daily tasks and users’ demand for the same. If there were more accountability for quality, however, vendors wouldn’t rush to market so soon with a buggy product. Quoting the article again: “‘Software is being treated in a way that no other consumer products are,’ said Barbara Simons, former president of the Association of Computing Machinery. ‘We all know that you can’t produce 100 percent bug-free software. But to go to the other extreme, and say that software makers should have no liability whatsoever, strikes me as absurd.’” Users are getting mad as heck, and they’re not going to take it anymore.

And so are developers and testers. How often have you been asked by management to cut corners? It is true that the natural tension between management and development is necessary to balance the forces that demand quality as well as profitability, but the nature of most organizations stacks the deck in favor of management in the wrong places. When that happens, quality takes a hit. A healthy organization will place management and development on a more equal footing as far as quality is concerned.

No document I’m aware of presents the case for moral principles in software development better than the code of ethics of the Association of Computer

Machinery (ACM) [3]. Here is a sampling of its tenets:

- Contribute to society and human well-being.
- Avoid harm to others.
- Strive to achieve the highest quality, effectiveness, and dignity in both the process and products of professional work.
- Acquire and maintain professional competence.
- Accept and provide appropriate professional review.

Software may well deserve its bad rap because of a lack of integrity in the organizations and processes that produce it. Frederick Brooks’ celebrated *Mythical Man Month* showcases the following quote from a fine French restaurant:

“Good cooking takes time. If you are made to wait, it is to serve you better, and to please you.”

We need to be up front about the cost of quality software. Managers need to trust developers when it comes to technical matters, and customers need to trust that vendors are conducting business honestly. And that trust must be earned.

Someday, after the political, legal, and economic dust has settled around the business of software, we may enjoy what famed physicist Richard P. Feynman envisioned in an address to Caltech graduates more than thirty years ago: “the good luck to be somewhere where you are free to maintain ... integrity ... and where you do not feel forced by a need to maintain your position in the organization, or financial support, or so on, to lose your integrity.” [4] {end}

## REFERENCES:

- 1) “Official Trapped in Car After Computer Failed.” Reuters, May 12, 2003.
- 2) “Spread of Buggy Software Raises New Questions.” CNN.com, April 27, 2003.
- 3) [www.acm.org/constitution/code.html](http://www.acm.org/constitution/code.html)
- 4) Quoted in *Feynman Lectures on Computation*. Perseus Publishing, 1996, p. 292.

IBM®







**Rational**

#### \_INFRASTRUCTURE LOG

\_DAY 90: Our software engineers are completely overwhelmed by this system we're building. It's so complex. The deadline is tight. Performance and quality can't be anything less than perfect. Everything depends on the software we're developing. It's all so...I don't know...huge.

\_DAY 92: The team is tired of typing one key at a time.

\_DAY 93: I've taken back control with IBM Rational Systems Development solutions. They'll give us the tools we need to manage our complex systems development through the entire lifecycle, from inception to deployment. Their disciplined approach looks at the full range of processes holistically.

\_We're all back to normal size again. Except Gil's brain. Which is still a bit on the tiny side.

Download the Systems Development e-Kit at:  
**IBM.COM/TAKEBACKCONTROL/SYSTEMS**

## EDITOR'S PICK

### Is Your Refrigerator Running?

"There ought to be a room in every house to swear in." ~ Mark Twain

When we moved into our new home, the refrigerator's ice maker was, naturally, disconnected. It's one of those Homeowner 101 things, so it was no surprise when I flipped the ice maker's arm down that the machinery failed to do anything at all.

Our new kitchen has excellent cabinet space—the sort of cabinets that leave no wasted space around the room and even wrap tightly around the refrigerator. This is one of those things you do when you like your refrigerator very much and either plan to keep it eternally or hope that the model will never cease production. For the new homeowner, it means that there's nowhere to get a good grasp on the thing to pull it out from its niche.

But my wife and I consider ourselves to be reasonably handy around the house, and it didn't take too long to work a system of ropes through the bottom of the fridge to move it. Once out, I slid behind the fridge and attached the copper water line to the back of the fridge.

Or, I would have, if the nut on the compression fitting hadn't been stripped.

What followed was a series of attempts, mistakes, self-loathing, fridge-loathing, trips to the hardware store, and, in the end, a phone call to a professional handyman. The compression fitting was no big deal, but when the copper pipe snapped off near where it enters the floor and I could find no access to its path through the recently finished basement, I had to admit that I was in a bit over my head. As I looked around the house at all the still-packed boxes, the surfaces left to clean, and the furniture that needed moving to its proper destination—and I realized that I'd spent most of a Sunday not connecting the ice maker—I began pondering my commitments.

Which brings me ever so conveniently to my Editor's Pick for this month, Michele Sliger's "In Search of Commitment Clarity." In it, Michele tells the story of a team member who overcommits to a heavy workload during the course of one iteration and pays the price. "We'll never get the product out the door if I don't push!" he tells his concerned manager.

There's something about being in a new house that makes one want to rush to see a long list of projects accomplished. But it's a new house. It isn't going anywhere, and neither are we—and neither are the projects. Those nifty, newfangled silicone ice trays have proven to be a worthy, temporary fix, and I'll commit to the ice maker when I have sufficient resources (read: when the handyman calls back).

Read "In Search of Commitment Clarity" at  
[www.stickyminds.com/editorspick10-6](http://www.stickyminds.com/editorspick10-6)



Joseph McAllister  
Managing Editor,  
Multimedia  
[jmcallister@sqa.com](mailto:jmcallister@sqa.com)

## Quotables

Nice assessment. The biggest trap that I've seen in analysis of performance data is summed up in an observation by Weinberg (for which I can't find the reference; sorry) that if you look at a small enough portion of any line, you'll see linear performance.

It's important to remind ourselves correspondingly that when we do see linear performance, we're looking at a portion of the line.

MICHAEL BOLTON COMMENTING ON DANNY FAUGHT AND REX BLACK'S "PEELING THE PERFORMANCE ONION"  
[www.stickyminds.com/quotables10-6a](http://www.stickyminds.com/quotables10-6a)

It's easy, especially at the beginning of a project, for a project team and the people who request deliverables (some of which are requirements) to be unable to differentiate between goals and requirements. The project team might be excited about the project and want to do everything. The people who want the release might feel as if there's pressure for everything in this release. But it's too easy for the project team members to be sidetracked if they haven't differentiated between goals and requirements.

JOHANNA ROTHMAN, "WHAT ARE YOU WORKING ON?"  
[www.stickyminds.com/quotables10-6b](http://www.stickyminds.com/quotables10-6b)

In the past, I've had managers with the "open door" policy, but when I poked my head in the door (which was open) and asked if they had a second, I got a furrowed brow and an annoyed, "Yeah."

Perhaps busy managers could help themselves by closing the door if they aren't prepared to have someone walk through it.

STICKYMINDS.COM MEMBER M GEORGE COMMENTING ON NAOMI KARTEN'S "OPENING THE DOOR TO BETTER OPEN DOOR POLICIES"  
[www.stickyminds.com/quotables10-6c](http://www.stickyminds.com/quotables10-6c)

Ever heard the phrase "The best defense is a good offense"? That is what this final principle of dynamic-path handling asserts.

By creating automated tests that attack the application in whatever state it exists, the tests will be more robust and, in many respects, more effective. Dynamic-path handling is a concept that is akin to model-based test automation. Model-based test automation is an automation approach in which the tests are derived by creating and implementing a model of the functional parts of the AUT. Dynamic-path handling, while similar, is smaller in scope. Where model-based test automation normally begs for a relatively complex framework, dynamic-path handling concepts may be implemented within any framework.

DION JOHNSON, "TAEKWONDO-MATION—TRAINING YOUR SCRIPTS FOR DYNAMIC COMBAT: PART IV—STRIKES"  
[www.stickyminds.com/quotables10-6d](http://www.stickyminds.com/quotables10-6d)



## Testing COTS: When, How, and How Much?

BY LINDA HAYES

Testing processes and practices are well defined and generally understood for internally developed applications, but what about those that are licensed from third parties? Granted, the vendor has responsibility for testing its own products, but the possibility of the software failing still exists and can be costly, even devastating; blaming others offers little consolation. If you rely on a commercial off-the-shelf (COTS) application, where does your trust in the vendor end?

[www.stickyminds.com/eLetter10-6a](http://www.stickyminds.com/eLetter10-6a)

## How Much Is Enough?

BY MICHAEL BOLTON

Exploratory testers design and execute tests in the moment, starting with an open mission and investigating new ideas as they arise. But how do we know when to stop? The first step is to recognize that we can't know when we're done, because any approach to answering the stopping question is necessarily heuristic.

[www.stickyminds.com/eLetter10-6b](http://www.stickyminds.com/eLetter10-6b)

## Test Notes and Coverage Maps—Aids for Rapid Testing

BY SRIDHAR KASIBHATLA AND ANDREW ROBINS

As delivery cycles get shorter, rapid test techniques are gaining in popularity. In this article, Sridhar Kasibhatla and Andrew Robins explore the concept of using coverage maps and test notes to support exploratory testing and concurrent test design. These maps and test notes also are used to review and track test coverage and can help document dynamically generated test cases for future reuse.

[www.stickyminds.com/eLetter10-6c](http://www.stickyminds.com/eLetter10-6c)

*A sampling of content from our eNewsletter archives*

**StickyLetter:** April 16, 2008

## The Way of the Foot and Fist

by Francesca Matteu

My gym membership has gone to waste, and I've traded my workout clothes for a dobok (a martial arts uniform). That's right: I've become a student of a local martial arts school and now don a white belt in taekwondo.

The environment within this new gym, which is called a dojan, is totally different from a regular gym. The dojan I visit is nothing more than one big rectangular room. The shortest walls are lined with mirrors. The main floor is padded, and the school's and the Korean flags are proudly displayed on the main wall. Kids and adults are everywhere, and it seems as if everyone is yelling all the time. It's definitely a cacophonous environment when you first enter.

But the raucousness ends as soon as the instructor demands your attention. It's very militaristic considering that if the instructor tells you to do something, you must answer with a loud, "Yes sir!" And then you must do as you're told, or else you'll be punished by having to do pushups.

Why on earth did I strip myself from a posh gym to become a taekwondo student? Because a gym, in my opinion and for what I want out of an exercise regimen, contains too much pomp for me.

The simplistic nature of taekwondo exercises is surprisingly refreshing. I don't have to worry about power sets or doing three different types of lunges to work out leg muscles. There are no weights in a dojan, just the instructor commanding you to do one hundred pushups in two minutes, of which I can only do half. Other strength-building exercises include crunches and squats. But the real training lies in learning punching, blocking, and kicking techniques.

To me this is like the perfect agile methodology to working out. Here's why: In a gym, you set some fantastic goal that you spend a year or several trying to achieve. In taekwondo, the ultimate goal is broken up into iterations in the form of belt tests. In a gym, you may or may not have a personal trainer who helps you use the equipment properly to help reach said goal. In the dojan, the instructor gives you constant feedback on your form and effort.

The biggest change is the culture. Gym members, for the most part, stick to themselves. But there's no retreat for the shy in a dojan. You practice every move with a fellow student.

As I mentioned, we work toward little goals. Even before your next belt test, you have to prove to the instructor that you're ready to test, which means taking many minor tests in class. In a regular gym, I "tested" against my body-mass index (BMI). But without constant supervision, I unfortunately didn't know if the length of my exercise routines was helping me achieve this goal or not. I soon cared less about my BMI and settled for an hourlong exercise free-for-all.

In taekwondo every learning step seems regimented, but ultimately you set the pace for advancement. If you fail a pre-test, then you know you have to go back and continue practicing that technique until you've mastered it.

There is something you should know about taekwondo: It has its advantages and disadvantages. For example, taekwondo is a good martial art as long as you maintain at least an arm's length between you and the opponent. Agile development also has its strengths and weaknesses. What do you think are some of the good points and bad? Email me your thoughts.

Until next time, live long, explore new ideas, and build better software.

[fmatteu@sqe.com](mailto:fmatteu@sqe.com)

## POWERPASS POINTER

### High Performance Testing

BY SCOTT BARBER

As an activity, performance testing is often misunderstood, especially by executives and managers. Try these tips on your next performance testing project and put your team on the fast track to success.

[www.stickyminds.com/eLetter10-6d](http://www.stickyminds.com/eLetter10-6d)



# ARE YOU IN SEARCH OF QUALITY? SO ARE WE.

Quality Assurance Analysts  
Quality Assurance  
Software Engineers

Apply online at  
[www.blackbaud.com/QAcareers](http://www.blackbaud.com/QAcareers)  
or contact Stephanie McDonald,  
senior technical recruiter, at  
**843.654.3547** or  
[stephanie.mcdonald@blackbaud.com](mailto:stephanie.mcdonald@blackbaud.com).

RELOCATION ASSISTANCE  
IS AVAILABLE

## Quality of Work

Work at the high-tech industry leader  
in software and services for nonprofits.

Use your talents to make a difference  
for our customers, who make a difference  
in the world.

Be an integral part of our product  
development team and help bring  
the best possible products to our  
customers.

## Quality of Life

Enjoy the history, culture, and charm  
of Charleston, South Carolina. Receive  
great benefits and a competitive salary.

Get started today!

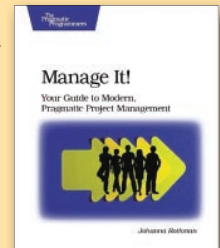
**Blackbaud®**

## Book Review

### *Manage It! Your Guide to Modern, Pragmatic Project Management*

by Johanna Rothman

Reviewed by Jennifer Cannon



Although I'm not a project manager, I am familiar with Johanna Rothman and have learned a lot from her insight and experience in software development, which she shares in her articles, papers, and books. This book is yet another example of her drawing upon her considerable experiences working with different companies and projects to share an incredible wealth of knowledge and practical advice for effective project management—in a way that is very easy to read. You will not be bombarded by a lot of theory here, but you'll be guided to project success by being presented with plenty of practical experience and advice from one of the best in the industry.

Rothman places an emphasis on the context of projects—there is no one correct process that will work for all projects and all lifecycles. Each project and each organization is different. One size does not fit all!

*Manage It!* covers all aspects of project management of software development projects. Rothman's emphasis is on agile projects, but if you don't use an agile lifecycle, don't worry. She provides tips and project management practices that are applicable for more serial lifecycles as well. Johanna begins the book with "Starting a Project" in which she helps you answer the following questions: What's driving my project? What does "done" really mean for my project? What does quality mean for my project? Also covered in this chapter is how to write a project charter and why this is so important.

Rothman then moves to planning the project, defining your lifecycle, scheduling, and estimating the work. She includes a chapter called "Recognizing and Avoiding Schedule Games" that covers several schedule games including "Pants on Fire," "Bring Me a Rock," and "Hope Is Our Most Important Strategy." Rothman defines how to recognize these games and provides advice for each that can be used to bring the project team and schedule back to reality and keep the project on track.

Johanna also covers creating a project team, project steering and rhythm, project dashboards, testing, and project completion. She has a chapter called "Managing Meetings" that is terrific! It's a fact that many project managers (and their team members) spend a lot of their time in meetings. Are they all really necessary? Rothman helps the reader decide what meetings to call, which ones to attend, which ones to delegate, and which ones to ignore. If you are a project manager with multi-site or multiple projects, then there is some advice in this book for you as well.

I highly recommend *Manage It!* to all project managers—from novices to those with more experience. This book is an incredible resource that should be referred to frequently for advice on how to help you decide which project management practice or technique is appropriate for your project—which she helps you to apply immediately. And she offers advice on how to avoid project pitfalls in order to steer your project to a successful completion.

# AGILE SOFTWARE DEVELOPMENT TRAINING

*Accelerate Your Career  
& Empower Your Team*

**NEW**

**FALL 2008  
SCHEDULE**

**BUILD-YOUR-OWN  
TRAINING WEEK**

Maximize the impact of your training by combining courses in one location to create a customized training week. Pair two courses and save up to \$300. For a complete list of courses available, visit [www.sqetraining.com](http://www.sqetraining.com) or call 888.268.8770 or 904.278.0524 for pairing discount options.

**SQE  
TRAINING**

[www.sqetraining.com](http://www.sqetraining.com)

Improve your skills and help your organization increase its performance through targeted high-value training. Delivered by top software consultants, training through SQE Training is one of the best investments you can make to meet your business objectives.

## AGILE SOFTWARE DEVELOPMENT TRAINING WEEK LOCATIONS

September 8-12, 2008	Washington, DC
September 22-26, 2008	Denver, CO
October 6-10, 2008	Chicago, IL
October 20-24, 2008	Seattle, WA
November 3-7, 2008	San Diego, CA

## Choose from 4 specialized training courses:

### THREE-DAY COURSES (Monday - Wednesday)

- Scrum Master Implementation Workshop
- Practical Test-Driven Development

### TWO-DAY COURSES (Thursday - Friday)

- Design Patterns Explained
- User Stories and Estimation in Agile Development



Let SQE Training come to you. For more information about on-site training courses, contact SQE Training at 904-278-0524 x212 or 888-268-8770 or email [onsitetraining@sqe.com](mailto:onsitetraining@sqe.com).



# Programming with GUTs

by Kevlin Henney

Looking around at some of the writing on the subject, both online and off, you could be forgiven for thinking that test-driven development (TDD) is just another way of saying *testing*, simply a synonym for *programmer testing*, or a fancy way of promoting plain ol' *unit testing*. It is not that these concepts do not have value—they do—it's that the arbitrary use of the term TDD decreases both the value of TDD and of other points of view. Not all testing is unit testing, and not all unit testing derives from TDD. As Alistair Cockburn noted in his blog earlier this year [1]:

Very many people say “TDD” when they really mean, “I have good unit tests” (“I have GUTs”?) Ron Jeffries tried for years to explain what this was, but we never got a catch-phrase for it, and now TDD is being watered down to mean GUTs.

Well, now at least there is a catch phrase for it! The distinction being drawn is both clear and useful: TDD is a way of doing something—i.e., it is a process; GUTs are an outcome—i.e., they describe an artifact and a quality of that artifact.

## What Do You Want From Your Tests?

Tests are commonly viewed in terms of offering quantitative feedback on the presence or absence of defects in specific situations. They are, however, in a position to offer much more than this.

For example, our tests can give us feedback on how loosely coupled our code is. Some portion of any nontrivial system is necessarily not unit testable: The portion of the code that must interact across the system boundary may be integration testable, but it is not unit testable. This is the code that may

```
public class RecentlyUsedList ...
{
    public RecentlyUsedList() {...}
    public void Add(string newItem) {...}
    public int Count
    {
        get {...}
    }
    public string this[int index]
    {
        get {...}
    }
    ...
}
```

Listing 1



ISTOCKPHOTO

be mocked out for purposes of unit testing other parts of a system, but should also be tested in its own right.

So, if there is some portion of our code that is necessarily not unit testable, how much should be unit testable that isn't? If the only code-facing tests we can write are integration tests, the chances are good that our code is too tightly coupled, no matter what dewy-eyed vision we may hold of its architecture. Furthermore, this also says something about the relationship between test quality and code quality: How good our set of unit tests can be is constrained by the quality of the production code's architecture.

For us to consider our tests good, we may want other things from them, such as coverage. However, because it's less familiar territory for many programmers, let's keep exploring qualitative rather than quantitative feedback. When we look at test code, do we expect to see something written only for machine execution, or is human consumption also a key consideration? In the blurb for their conference session, “Are Your Tests Really Driving Your Development?” [2], Nat Pryce and Steve Freeman make clear the importance of the human audience:

Everybody knows that TDD stands for Test Driven Development. However, people too often concentrate on the words “Test” and “Development” and don't consider what the word “Driven” really implies. For tests to drive development they must do more than just test that code performs its required functionality: they must clearly express that required functionality to the reader. That is, they must be clear specifications of the required functionality. Tests that are not written with their role as specifications in mind can be very confusing to read.

```
[Test]
public void Test()
{
    RecentlyUsedList list = new RecentlyUsedList();
    Assert.AreEqual(0, list.Count);
    list.Add("Aardvark");
    Assert.AreEqual(1, list.Count);
    Assert.AreEqual("Aardvark", list[0]);
    list.Add("Zebra");
    list.Add("Mongoose");
    Assert.AreEqual(3, list.Count);
    Assert.AreEqual("Mongoose", list[0]);
    Assert.AreEqual("Zebra", list[1]);
    Assert.AreEqual("Aardvark", list[2]);
    list.Add("Aardvark");
    Assert.AreEqual(3, list.Count);
    Assert.AreEqual("Aardvark", list[0]);
    Assert.AreEqual("Mongoose", list[1]);
    Assert.AreEqual("Zebra", list[2]);
    bool thrown;
    try
    {
        string unreachable = list[3];
        thrown = false;
    }
    catch (ArgumentOutOfRangeException)
    {
        thrown = true;
    }
    Assert.IsTrue(thrown);
}
```

## Listing 2

The difficulty in understanding what they are testing can greatly reduce the velocity at which a codebase can be changed.

This consideration is perhaps more explicit and more obviously enabled with TDD, but the sense and sensibility of treating tests as specs are also relevant when using other unit-testing approaches.

## Style and Substance

Consider a simple example: a recently used list, which is a collection that holds strings uniquely and in reverse order of their insertion. The C# code in listing 1 shows the key features for using such a class: a default constructor, an Add method, a Count property to query size, and an indexer for subscripting.

So what style should our tests adopt in order to communicate functionality to the reader? The NUnit test case in listing 2 certainly exercises the class, but lumping all test logic into an undifferentiated, monolithic slab called `Test` has little communication value. It can be justified for code with simple behaviors, but it does not scale well to larger tests.

Dividing up the single test method into a number of arbitrary test methods—For example, `Test1`, `Test2`, and

```
[Test]
public void Constructor()
{
    RecentlyUsedList list = new RecentlyUsedList();
    Assert.AreEqual(0, list.Count);
}

[Test]
public void Add()
{
    RecentlyUsedList list = new RecentlyUsedList();
    list.Add("Aardvark");
    Assert.AreEqual(1, list.Count);
    list.Add("Zebra");
    list.Add("Mongoose");
    Assert.AreEqual(3, list.Count);
    list.Add("Aardvark");
    Assert.AreEqual(3, list.Count);
}

[Test]
public void Indexer()
{
    RecentlyUsedList list = new RecentlyUsedList();
    list.Add("Aardvark");
    list.Add("Zebra");
    list.Add("Mongoose");
    Assert.AreEqual("Mongoose", list[0]);
    Assert.AreEqual("Zebra", list[1]);
    Assert.AreEqual("Aardvark", list[2]);
    list.Add("Aardvark");
    Assert.AreEqual("Aardvark", list[0]);
    Assert.AreEqual("Mongoose", list[1]);
    Assert.AreEqual("Zebra", list[2]);
    bool thrown;
    try
    {
        string unreachable = list[3];
        thrown = false;
    }
    catch (ArgumentOutOfRangeException)
    {
        thrown = true;
    }
    Assert.IsTrue(thrown);
}
```

## Listing 3

`Test3`—does not really address the issue. Such a division and labeling is, well, arbitrary, so it does not communicate anything useful to the reader.

Programmers who have moved beyond the monolithic and arbitrary styles tend to gravitate toward a procedural style. A procedural style can be characterized in terms of “I have a procedure `foo`, therefore I have a corresponding test procedure that tests `foo`.” Listing 3 shows procedural unit tests for our `RecentlyUsedList` class, each test method aligned with a method in the class under test.

```

[Test]
public void InitialListIsEmpty()
{
    RecentlyUsedList list = new RecentlyUsedList();

    Assert.AreEqual(0, list.Count);
}
[Test]
public void
AdditionOfSingleItemToEmptyListIsRetained()
{
    RecentlyUsedList list = new RecentlyUsedList();
    list.Add("Aardvark");

    Assert.AreEqual(1, list.Count);
    Assert.AreEqual("Aardvark", list[0]);
}
[Test]
public void
AdditionOfDistinctItemsIsRetainedInStackOrder()
{
    RecentlyUsedList list = new RecentlyUsedList();
    list.Add("Aardvark");
    list.Add("Zebra");
    list.Add("Mongoose");

    Assert.AreEqual(3, list.Count);
    Assert.AreEqual("Mongoose", list[0]);
    Assert.AreEqual("Zebra", list[1]);
    Assert.AreEqual("Aardvark", list[2]);
}
[Test]
public void
DuplicateItemsAreMovedToFrontButNotAdded()
{
    RecentlyUsedList list = new RecentlyUsedList();
    list.Add("Aardvark");
    list.Add("Mongoose");
    list.Add("Aardvark");

    Assert.AreEqual(2, list.Count);
    Assert.AreEqual("Aardvark", list[0]);
    Assert.AreEqual("Mongoose", list[1]);
}
[Test, ExpectedException(ExceptionType =
typeof(ArgumentOutOfRangeException))]
public void OutOfRangeIndexThrowsException()
{
    RecentlyUsedList list = new RecentlyUsedList();
    list.Add("Aardvark");
    list.Add("Mongoose");
    list.Add("Aardvark");
    string unreachable = list[3];
}

```

Listing 4

Although there is a certain logic to procedural testing, the rationale is somewhat weak (even for procedural code). Testing individual methods in isolation doesn't really make sense in terms of what a `RecentlyUsedList` is and how it is used. In many cases it is impossible to focus on just one method without involving others. For example, we cannot call the `Add` method without also having executed a constructor. Likewise, even in the constructor's test, we still end up querying the `Count` property in order to say something meaningful about the result of construction.

A per-method testing style also leads to an uneven spread of test-case length and depth. For example, most of what a `RecentlyUsedList` is about is covered by the `Add` method. A single test case for the `Add` method slops a number of different usage scenarios into one bucket. Indexing is also somewhat important, but unless we use `Add` we cannot usefully test the indexer. Indeed, much of the testing of `Add`'s behavior ends up in the indexer's test!

We can do better. A good unit test needs both to illustrate and define the behavioral contract of the unit in question. Behavior is more than just individual methods, so we need a style that cuts across the interface, a style where each test case is structured in terms of a meaningful and specific behavioral goal, as shown in listing 4.

One of the more noticeable consequences of a behavioral style is the naming. The idea is that each method is named as a requirement and therefore provides an outline of the contract:

- Initial list is empty.
- Addition of single item to empty list is retained.
- Addition of distinct items is retained in stack order.
- Duplicate items are moved to front but not added.
- Out-of-range index throws exception.

Compare this rich description with the somewhat simplistic procedural summary:

- Constructor
- Add
- Indexer

As an aside, a behavioral style forms one cornerstone in behavior-driven development, which Dan North described in his *Better Software* article "Behavior Modification" [3].

In comparing the monolithic, arbitrary, procedural, and behavioral styles, we can see there is a great deal more to good unit tests than simply mastering the syntax of an assertion.

**{end}**

#### REFERENCES:

- 1] Cockburn, Alistair. "The modern programming professional has GUTs." [alistair.cockburn.us/index.php/The\\_modern\\_programming\\_professional\\_has\\_GUTs](http://alistair.cockburn.us/index.php/The_modern_programming_professional_has_GUTs).
- 2] Pryce, Nat; Freeman, Steve. "Are Your Tests Really Driving Your Development?" [xpdlay6.xpdlay.org/programme.php#Are\\_Your\\_Tests\\_Really\\_Driving\\_Your](http://xpdlay6.xpdlay.org/programme.php#Are_Your_Tests_Really_Driving_Your).
- 3] North, Dan. "Behavior Modification." *Better Software* magazine. March, 2006.



**What factors and styles have you seen that have influenced the quality of automated tests?**

Follow the link on the **StickyMinds.com** homepage to join the conversation.





# World Forum Convention Center

The Hague, Netherlands  
10<sup>th</sup> - 13<sup>th</sup> November

## "The Future of Software Testing"

EuroSTAR is Europe's Largest Software Testing Event with over 1200 attendees in 2007. EuroSTAR provides a valuable opportunity for you and your test team to avail of the very best and most intensive testing training week in Europe. Join us in the Hague in November where you can participate in...

- **5 World Class Keynote Presentations**
- **14 Intensive Tutorial Training Courses** delivered by some of Europe's greatest Test Experts
- **37 Track Sessions** covering the complete range of software testing topics about the Future of Software Testing, Testing in Agile Environments, Model-Based Testing, Outsourcing, Case studies, Test Environments, Metrics, Test Techniques, Test Management, Performance Testing and many others
- **4 Mini-Tracks** which give an opportunity for less experienced speakers to make their first contribution to the profession by sharing their experiences.
- **2 Participatory Workshops** based on research into Heuristic Reasoning and the new ISO 29229 testing standard
- **3 Manifesto Workshops** During the conference special workshops will create a statement for the testing community worldwide, about the future of software testing. This will take the form of a Manifesto that gives the major guidelines for our journey into the future
- **World's Largest Test Tools and Services Exhibition** featuring the most innovative and integral companies in the European testing industry
- **The EuroSTAR Gala Awards Evening** where the greatest Testing Achievements in 2008 will be recognized
- **Networking Opportunities** Meet with speakers, sponsors, exhibitors and your test colleagues to share ideas and shape 'The Future Of Software Testing'

Visit the website to view the full conference programme

[www.qualtechconferences.com](http://www.qualtechconferences.com)



## Keynote Speakers



### The End of Testing As We Know It

**James Whittaker,**  
Microsoft, USA



### Trends That May Shape the Future of Software Testing

**Randall Rice,**  
Rice Consulting Services Inc., USA



### Test Maturity Model Integration (TMMi) - Process Improvement for the Present and Future

**Erik Van Veenendaal,**  
Improve Quality Services BV,  
The Netherlands



### Becoming Agile - Reshaping Testing for an Agile Team

**James Lyndsay,**  
Workroom Productions Ltd., UK



### Testing on the Toilet: Revolutionising Developer Testing at Google

**Bharat Mediratta and**  
**Antoine Picard,**  
Google, USA

# Two Cheers for Ambiguity

by Michael Bolton

I was sitting at the back of the room, munching on a donut, sipping a coffee, and listening to the presenter talking about the importance of unambiguous requirements. “What does he mean by ‘unambiguous?’” I wondered.

He also seemed to be opposed to jargon—yet, jargon, to those who use it, is extremely precise and unambiguous. “Jargon” and “ambiguity” are like “quality” or “purpose”—subjective and context-dependent, not properties of something but rather a relationship between some person and the thing. Ambiguity may be a problem or it may not, depending on its meaning and significance to some person.

I was reminded of all this recently when a colleague observed that he avoided using words like “skill,” “diversity,” “problems,” and “mission,” because he found them inherently ambiguous. I replied that I use these words constantly for exactly the same reason. I find them to be not only necessary but also useful as gauges for assessing consensus on a project and how we get to that consensus.

Some people are comfortable with ambiguity; others are not. We can spot a member of one group or the other by the answer we get when we ask, “Can we talk about what you mean by ‘skill’ (or ‘diversity’ or ‘test’ or ‘bug’)?” When someone responds, “Well, for example ...” or “In this context ...” or “There are many possible meanings, but around here we mean ...” then we know we’re in good company. We can have an evolving, ongoing conversation that helps us work together and understand one another, and if we discover that we don’t have a consensual understanding of something, we can work it out. On the other hand, when someone responds, “Isn’t that obvious?” or “Why do you keep going meta?” or “Can’t we talk about practical stuff?” then we know that there’s work to be done, because someone is suffering from that terrible disease, Single Model Syndrome.



DREAMTIME

Single Model Syndrome is the silver bullet for ambiguity problems; something is unambiguous when there’s no possibility of a second interpretation of it. On the other hand, Single Model Syndrome can lead to frightful misunderstanding, especially when two people suffering from it—and using different models—show up at the same meeting.

The battle against ambiguity has to do with the problem of closure. Psychologically, some people are comforted by closure and require it; others don’t require it and, in fact, may be leery of it. Developers and project managers tend to value closure because it gives them a finish line, a clear goal that can be met. Good testers are aware of the risk of closure, especially when it’s premature. Suspending conclusions helps us to see more alternatives and to adapt to change, both in problems and in solutions. Seizing certainty at the requirements stage cuts us off from alternative approaches and new information. One common manifestation of this problem is an excessively detailed test plan—one that doesn’t match the product that is eventually delivered.

For testers in particular, *recognizing* ambiguity is useful. Recognizing ambiguity is naturally important because ambiguity is a way in which misunderstanding may provide homes for bugs in the product. Yet ambiguity, which

implies more than one possibility, might also be a blessing in disguise. An ambiguous sentence might trigger a discussion about what we perceive, what we agree upon, and what we don’t yet understand. An ambiguous word might have several open-ended meanings that help reduce tunnel vision. An ambiguous problem statement might remind us that there are often several alternative approaches to solving a problem. The precise expression of a requirement *might* make testers’ lives easier, but perhaps the meaning and the significance of the requirement are more important, even though they may be imprecise.

James Bach tells a wonderful story that illustrates the distinction. On a project several years ago, a junior tester asked James to help interpret a line in the requirements document that said, “When the user presses the touchscreen, the system shall respond within 300 milliseconds.” Holding a stopwatch in one hand and using the system with the other seemed impractical, and automation seemed to have a high development cost, so James decided to train the testers to recognize 300 milliseconds. He bought an inexpensive stopwatch for each of the testers. They went to lunch and practiced turning their stopwatches on and off until they could estimate 250 milliseconds, plus or minus fifty, with rea-

sonably good precision. Then someone brought up the question, “What if we were off by ten milliseconds, and the system took 310 milliseconds to respond? That would be a failure, but would that be a *problem*?”

James realized that he had considered the precise wording of the requirement—a shallow sort of meaning—but neither its deeper meaning nor its significance. He went to the project manager for clarification. It turned out that the previous version of the program had taken upward of seven seconds to acknowledge that it had received input, and customers had found this annoyingly slow. The designers had put the precise timing—300 milliseconds—into the specification because they had thought that you couldn’t use words like “annoyingly slow” with testers. James suggested instead that the developers show the testers the old system so the testers could understand the problem from a visceral perspective. In this case, ambiguity was disguised as precision—and this was no place for stopwatches.

So how should we deal with ambiguity in requirements and elsewhere in the project? How do we seek it, and how do we resolve it? Here are some heuristics.

The first thing is to recognize that the *requirements* are not the *requirements document*; at best, the document is a stand-in for the ideas of one or more real people. Recognize that all statements, whether written or spoken, are potentially ambiguous, but the ambiguity might not represent a problem for the project, so look for ambiguity that *matters*. A testable requirement is not necessarily one that is painstakingly precise, mathematically falsifiable, or unfailingly unambiguous. A testable requirement is one that helps us ask and answer the question “Is there a problem here?”

Conversation is a fast and powerful approach to discovering and resolving ambiguity. Ask plenty of questions and watch for disagreements on the answers from various people; then seek consensus on meaning and significance. There can be several levels to the conversation—one in which we’re talking about something, another in which we ensure that we agree on what we’re talking about,

and yet another in which we work out a protocol for resolving our differences. This may seem like extra work but, in fact, people are doing it all the time. The trick is to do it *consciously*.

Make sure that conversations are supplemented by a wide variety of media—whiteboards, tables, scenarios, mind maps, wikis, knowledge-crunching sessions, diagrams—in addition to the more traditional forms of documentation. Be skeptical that any one document will identify all the things you need to know about your project.

When you spot ambiguity problems, make the problem clear by pointing out alternative interpretations: “There could be a bunch of testing missions in play here—finding important problems quickly, investigating the problems we’ve found, assessing backward compatibility, identifying new risks. What can we agree on as the primary goal?”

Don’t feel obliged to document minute details of every discussion. Documentation may have high cost and low value when consensus is the goal. Some

things on a development project are so important that we *don’t* write them down; instead, they become part of the project culture. (Everyone remember to breathe!)

Above all, remember Jerry Weinberg’s definition of a tester—a definition that highlights the significance of ambiguity in our work: “A tester is someone who knows that things can be different.”

{end}

**How do you know what you know? How do you know it? And how do you know that others understand things the same way?**

Follow the link on the [StickyMinds.com](http://www.StickyMinds.com) homepage to join the conversation.

### Sticky Notes

For more on the following topic go to [www.StickyMinds.com/bettersoftware](http://www.StickyMinds.com/bettersoftware).

■ Further Reading



**RALLY**  
SOFTWARE

Scaling Software Agility

## Rally is the #1 partner for Agile success

- Shorten development cycles
- Increase visibility and collaboration
- Synchronize global development teams



**Sign up for FREE Community Edition**

**Rally’s award-winning Agile life cycle management tool for a single team!**

[www.rallydev.com/bsm](http://www.rallydev.com/bsm)

**Jolt Product Excellence Award**  
2006 + 2007 + 2008  
**WINNER!**



# Going on a Picnic with James Watt

by Clarke Ching

“Takeout, please,” I said. It was a nice day; we’d sit outside and enjoy the sun. I looked around the coffee shop as I waited for the barista to do his thing, but Dave wasn’t here yet. It wasn’t like him to be late.

Thankfully, the barista was in no rush and Dave arrived three minutes later, just before the coffees did. I handed him his coffee and we made our way outside, across the road, and into the park where we found a comfortable bench.

“How are things?” I asked Dave.

“Absolutely horrible,” he said.

That’s strange, I thought. Dave is managing a software development project for one of CCXSoft’s big banking customers. It’s his first big project, and I’ve been coaching him since he started the commercial negotiations a few months ago. Things seemed to be going well four weeks ago when we last met, but the project hadn’t started yet.

I asked Dave how things could have turned so bad just three weeks into the project.

Dave let out a long, deep sigh. “Simple,” he said. “The customer has reneged on its commitments. The bank promised me its subject matter experts and testers would be on board two weeks ago at the very latest, but they’ve not arrived. The customer keeps promising the SMEs and testers are going to arrive any day now, but I’ve been hearing that for the past three weeks. I don’t believe it any more.”

I sipped my coffee silently, leaving Dave plenty of space to tell his story.

“I’m convinced that the project is going to run late, and I can’t do anything about it. It’s a mess.”

“You used the ‘picnic principle,’ right?” I asked.

The picnic principle states that if you are planning a picnic with a group of friends and you want them all to turn up, then you make sure that each person provides something unique and vital to

the picnic’s success. Provided everyone’s individual responsibilities are clear and everyone knows that if he doesn’t turn up then a vital part of the picnic will be missing and he’ll be letting down all of his friends, then it’s likely everyone will turn up; no one wants to be blamed for ruining the picnic. The same applies in projects.

“Sure. We planned this project together, and I’ve made it very clear to the customer that its SMEs and testers are vital to the success of the picnic.”

“The customer knows your team will be guessing at requirements without its help and, as a consequence, you will end up delivering an inferior product?”

“Absolutely. Garbage in, garbage out. I’ve repeatedly spelled this out to Mary, the bank’s IT project manager, and she says she knows this but the people we need don’t work for her. They work in other parts of the bank. She says she’s been repeating that exact message to them: If they don’t provide the right people, they’ll get a weaker product.”

“Mary keeps reassuring me that she is doing her best. The problem is that the SMEs and testers are working in operational jobs and their current bosses won’t release them for our project. Not yet, anyway.”

“It sounds as if Mary won’t feel all that much pain if they don’t turn up. Is that right?” I asked.

“Oh, she comes across as genuinely embarrassed by this. She keeps saying things like ‘Don’t shoot me, I’m just the messenger,’ and I’m quite sure she’s noted the delays prominently in her internal weekly status reports.”

I smiled. I’d dealt with this bank—and many companies just like it—and the successful managers all seemed more focused on protecting themselves from

being blamed for failure rather than ensuring success.

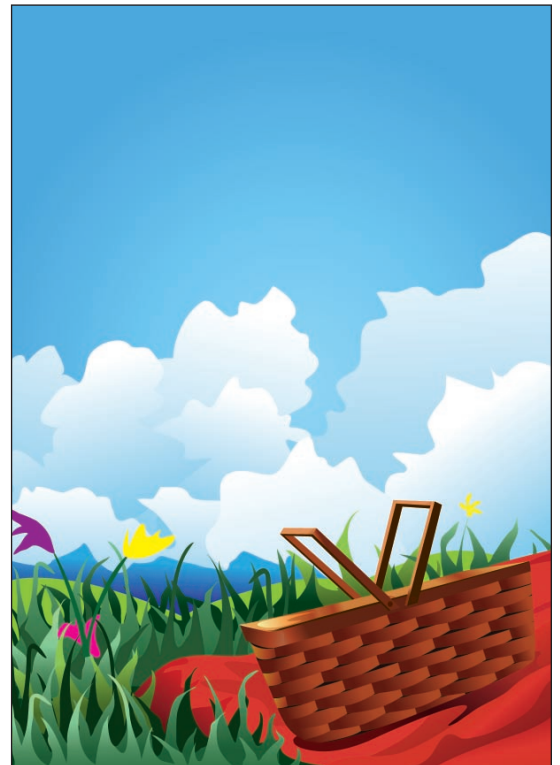
“So why are you bothered, Dave? You’re not going to be blamed for their internal problems. If they’re willing to take a lesser product, surely that’s their choice.”

Dave put down his coffee, thought a minute as he chose his words, then turned to face me.

“It bothers me because although we can deliver the product on the date we promised, that product then has to be accepted by the very same SMEs who should be working on the project NOW. They’ll reject the software because it doesn’t do what they want it to do. We’ll start fighting about who pays for the changes. We’ll lose money—unnecessarily—and we will have to delay starting work on other projects while we do the rework.”

“Rework that could be prevented if they’d just kept to their promise?”

“Precisely. Look, Steve, when I took on this project I promised our VP that I would show him how I could use the stuff we learned last year not just to satisfy but *delight* the bank. I don’t want merely to deliver what we agreed in some sterile contract. I don’t want the bank saying, ‘Well, the product is OK’



ISTOCKPHOTO

and that they ‘got the best they could expect.’ I want to delight them.”

Dave leaned forward. “Sure, I want them to get a good product, and I want to deliver on time, but it’s more than that. When Mary, the SMEs, and the testers leave this project, I want them to say, ‘Wow! What a great experience.’ I want them to thank us for our hospitality, to say they just loved the picnic. I want them to volunteer the next time their bosses want to do a project with us.”

Dave was talking faster now and his hands were flying over the place as he spoke.

“But, it’s even more than that. I need their guys on board, collaborating with my guys. When the SMEs meet with their bosses and are asked how things are going, I want them to be able to say that we’re working hard—that we are hustling, even. They can’t say that if they’re not here working with us. I want the bank to feel that it’s getting good value for its money. I can tell them that ... but I’d rather show them, each and every working day.”

I dropped my poker face and smiled. “So, what’s missing, then? What would make the difference?”

“Isn’t that obvious? I need their SMEs and testers on board, like they promised.”

“So, what’s missing, then? What would make the difference?”

He frowned, not sure what I meant. I gave him time.

“The folks I’m dealing with are willing to accept a lower standard than they need to.”

“So, what’s missing, then? What would make the difference?”

“What would make the difference? If I could just put my case to someone who cared. Someone who could do something about it.”

I said nothing. Dave knew what he had to do.

“It’s obvious, isn’t it? I have to talk to Mary’s boss—the project’s sponsor. I have to make him feel pain. Mary’s heart is in the right place, but she doesn’t have the authority to make people move faster. Her boss does.”

I nodded. He’d been having the right

## STORY LINES

- **Use the “picnic principle” when planning projects: Agree, with no ambiguity, who brings what to the project. Ensure everyone knows the consequences to him and to the project of not turning up. No one wants to ruin the picnic.**
- **Don’t just deliver what’s in the contract—deliver an experience. Your goal is not just to have a picnic, but for everyone to enjoy it.**
- **Google “James Watt Steam Engine.” You’ll learn that great technical innovations must also be great commercial innovations if they’re to survive.**
- **When people aren’t delivering on their promises, dollarize their pain. Sometimes numbers speak louder than words.**

conversations—just with the wrong person.

“Good, Dave. Now what are you going to say to him? You’ve described the benefits to you and CCXSoft for getting its people on board now ... but he’ll just laugh at you if you repeat those to him.”

Dave frowned. “I’m going to need to talk dollars, aren’t I?”

“It’s easier than you think. Let me tell you about Watt.”

“What?”

“No, Watt. James Watt—he helped kick start the industrial revolution. He also invented the term ‘horse power.’”

Dave raised his eyebrows slightly then smiled. Another one of my analogies.

“Watt was a clever technician and an astute business man. He didn’t invent the steam engine—he took an existing invention and improved it so it was commercially viable. After he’d done the innovative technical stuff, though, he found himself with a brand new problem: how to sell his new innovation. He decided to sell the steam engine to coal mines where miners could use it to replace the horses they used to pull coal and pump water from the mines. Watt needed to figure out how to sell to the owners of the coal mines, so he invented the term ‘horse power’ to describe how many horses each engine would replace. So a twenty-horse-power engine would replace twenty horses, saving the coal mine owners a lot of money. Watt took one third of the savings as his price, and both sides in the bargain came out substantially better off.”

“The bank doesn’t employ horses.”

“No, but imagine if you had the conversation with the sponsor where you say something like: ‘If your people don’t

come, then we will still deliver to you in time to start your acceptance testing, but we expect that it’ll take between two and four months longer for your team to do its acceptance testing. Some changes are going to come out of that—changes that we could avoid if your team were on board now. I estimate we will have to charge you between X and Y dollars to complete those. It’ll also mean that you will go live two to four months later than you currently plan. I imagine the delay will cost you considerable revenue, too. I’d like your help to prevent those costs.”

I looked at Dave as his frown slowly turned into a smile.

“Yes! That would work. He’s going to feel pain—that’s inevitable. He’s also got the power and authority to avoid it. But I’m not going to talk to him.”

“You’re not?”

“No. I’ll have the conversation with Mary first, then suggest we both go talk to her boss. Together we can come up with a stronger argument. And, more importantly, I want to collaborate with her on this project. How’s she going to feel if I go to her boss behind her back? That’s not collaborative ... that’s adversarial. And that’s no way to delight a customer.”

Good work, Dave. {end}



**Dave is right: Projects deliver much more than what’s in the contract, they also deliver an experience. Tell us about a great experience you’ve had and what made it great.**

Follow the link on the [StickyMinds.com](http://StickyMinds.com) homepage to join the conversation.





**HOW TO**

**FAIL**

**WITH**


**AGILE**

**20** Tips to Help You  
Avoid Success

BY CLINTON KEITH & MIKE COHN

ISTOCKPHOTO





**A**gile processes are now accepted as valid alternatives to traditional software development processes. Most people who adopt agile do so to realize the benefits of faster delivery, higher quality, products that more closely match user needs, and so on.

Not everyone is so enamored of agile. Some teams and individuals balk when a mandate to “become agile” is passed down from some “higher-up” in the organization or when some young go-getter decides to start an idealistic grassroots movement to effect change. A switch to agile often conflicts with personal goals such as maintaining the status quo, avoiding career risk, working no harder than necessary, or maintaining a large fiefdom of direct reports. It is to these individuals—those who have to become agile but don’t want to—that we would like to direct our advice. Don’t worry. We’re not going to try to seduce you into trying agile, convince you of its merits, or tell you how to succeed. No, we’re going to help you fail with agile. Then you’ll be done with it and can go back to your comfort zone.

Although there are many ways to sabotage your agile project, for convenience we have grouped them into four categories: management issues, team issues, product owner issues, and process issues. In each instance, we will cite an example of someone who successfully caused an agile adoption to fail, list the general guidelines for failure that the example is meant to demonstrate, and then list alternative techniques you can try to help you replicate the process. We hope this approach will allow you to fail quickly and avoid potential success.

### Management Issues

Drew had seen management fads come and go. In his mind, agile was no different. A quick learner, he read a number of books and even took a class on agile. He didn’t trust it, but, as a team player, it was his obligation to give it a try.

Drew picked team members and told them to “be agile.” He told them that

they would need to meet daily, estimate their work, and produce versions of their product (a database tool for storing artwork) every month.

Since Drew didn’t trust agile or his team’s ability, he attended every daily scrum, paying close attention and pointing out what the team was doing right and what it was doing wrong. Soon the daily meetings became a model of brevity and procedural correctness. As a bonus, no one spoke up about problems—especially in front of Drew. Drew had successfully followed our first guideline to agile failure.

**GUIDELINE 1:** *Don’t trust the team or agile. Micromanage both your team members and the process.*

To no one’s surprise, the team did not produce impressive results. It didn’t meet all of its iteration goals and was no more productive than it had been before. Drew conducted retrospectives that did not reveal any problems that he could fix. As a result, Drew threw away all the books he had read and directed the team to return to the old way of developing the project. Drew was following guideline 2.

**GUIDELINE 2:** *If agile isn’t a silver bullet, blame agile.*

While Drew went to one extreme by micromanaging his team, it is equally effective to go to the opposite extreme and not provide any guidance at all. Remember: While self-organizing agile teams are also self-managing, they are not self-leading. An agile team needs the type of leadership that provides a vision to work toward and motivation for achieving that vision. A strong agile leader, often in the form of a product owner, knows how to motivate a team with a description of an extremely desirable product that is just beyond what the team may think it can do. Freed to

pursue that goal and provided with ongoing guidance from a product owner, an agile team can become truly high performing. Don't give your team that opportunity! If micromanagement isn't your style, follow guideline 3.

**GUIDELINE 3:** *Equate self-managing with self-leading and provide no direction to the team whatsoever.*

While support for using agile may come from the highest levels of a company, often the adoption of agile will be driven by the team itself. Don't worry. You still have plenty of opportunities to create failure in those cases, especially if you are the manager. You may want to start by undermining the evangelist on the team—the one who has read all the books and is taking the chance to promote agile. Brush off the rules he is asking you to follow. Interrupt the daily scrum with new directions. Change the priority of the iteration goals. It works well and is encapsulated in guideline 4.

**GUIDELINE 4:** *Ignore the agile practices. They don't apply to management.*

If you want to be sure that agile doesn't take root, go straight to the team members themselves and let them know you think agile is a fad. Some of them will be skeptical to begin with, so it won't take much to convince them to ignore the practices. Remember, like Barney Fife, you have the power to nip this thing in the bud. Just follow guideline 5.

**GUIDELINE 5:** *Undermine the team's belief in agile.*

## Team Issues

Not all of us are managers. Don't worry, non-managers can wreak havoc at the team level, as well. Just take the case of the NotQuite team, tasked with developing inventory-management software. This team shows the power of consistency in bringing down an agile project. For its first iteration, NotQuite committed to completing six items from the product backlog; it finished four. Because it was the first iteration and most teams overcommit in their first iteration, the product owner cut the team a little slack. This didn't faze the NotQuite team. For the second iteration the team again planned to finish six items; it finished five. The slight improvement only

lulled the product owner into a false sense of security. NotQuite continued to chronically overcommit, falling short in the third, fourth, and fifth iterations. Soon, the product owner learned not to trust the team, and this undermined any success it may have had with agile—a fantastic implementation of guideline 6.

**GUIDELINE 6:** *Continually fail to deliver what you committed to deliver during iteration planning.*

When falling short, don't make the mistake of going all-out on every iteration, reaching the last day panting with exhaustion time after time. A team like that could almost be forgiven for never quite delivering what it planned. A key to NotQuite's failure was its cavalier attitude toward missed commitments. Team members made it clear that it really didn't matter if something was finished on the last day of the iteration (as had been committed) or a few days into the next iteration. What's a few days between friends? Remember, a few days here and there can add up to quite a lot. If a team continually misses its commitments, it makes it impossible for the product owner to make plans and external commitments. This leads to guideline 7 for how to fail at agile.

**GUIDELINE 7:** *Cavalierly move work forward from one iteration to the next. It's good to keep the product owner guessing about what will be delivered.*

Perhaps the best way to cause an agile project to fail is to follow guideline 8.

**GUIDELINE 8:** *Do not create cross-functional teams. Put all the testers on one team, all the programmers on another, and so on.*

Merrilynn was able to use this guideline to kill her company's pilot agile project. Her organization was developing an application that would have separate Windows and Web-based clients. As a development director, Merrilynn had control over team composition and was able to create three separate teams: a Windows team, a Web team, and a test team. This team structure worked against the goals of agile. If Merrilynn had wanted to succeed, she would have instead created three teams that each included Windows, Web, and test skills. Because Merrilynn kept the teams separate, she made it impossible for any

team to deliver the working software that an agile team is expected to deliver at the end of each iteration. Nicely done, Merrilynn.

Another option open to Merrilynn was putting all twenty of her people on one team. This would have violated the standard agile advice of creating teams of five to nine people. She could have justified it to anyone who questioned the decision by stressing the unique two-client nature of her team's product. If she had chosen to create one large team instead of three reasonably sized teams, Merrilynn would have substantially increased the communication overhead of her team. This would have slowed progress and created complaints about how all the conversations in agile were a tremendous burden. If separating teams is too hard to justify, you can bog down a project very easily by following guideline 9.

**GUIDELINE 9:** *Large projects need large teams. Ignore studies that show productivity decreases with large teams due to increased communication overhead. Since everyone needs to know everything, invite all fifty people to the daily standup.*

## Product Owner Issues

If the management and team guidelines aren't available to you, there is another route to take: Consider a take-down from the product owner angle. A product owner has many options at her disposal to bring an agile project to its knees. Take Kathy, for example, who was the product owner for a team working on a video game. The team was making great progress on features with every iteration and showing more player "fun" every time. Kathy let team members keep thinking that this was all they needed to do. She never attended reviews, rarely tried the game, and requested stories that were meant to steer the game toward the product she imagined. If that weren't enough, Kathy didn't share her own vision with the team or the other customers to whom she reported (such as marketing). A year into development, the game was demonstrated to a group of executives who were shocked at the direction the game had taken. It was not what they wanted to market. The disconnect between Kathy, the team, and

**“As you can see, failure at the product owner level is easily achieved through miscommunication, general ignorance of the team’s progress, and lack of education.”**

senior management caused the project to lose six months of progress. Well done, Kathy!

Kathy demonstrated several guidelines for how an agile project can fail at the hands of a product owner.

**GUIDELINE 10:** *Don’t communicate a vision for the product to the team or to the other stakeholders.*

**GUIDELINE 11:** *Don’t pay attention to the progress of each iteration and objectively evaluate the value of that progress.*

**GUIDELINE 12:** *Replace a plan document with a plan “in your head” that only you know.*

One of the tenets of iterative development is the discovery of the value of features being added as part of the whole. This is the reason that every iteration produces a potentially shippable release of the product. This is in stark contrast to plan-driven projects, which attempt to predict the utilization of resources so the product emerges complete from all the separate parts only at the end. When a product owner does not make that change, the team can quickly fail. It is just as critical to educate the product owner as it is to educate the team. Generally speaking, if you want to fail quickly, avoid training at all.

The crucial role of product owner often is balanced by someone else who acts as the team’s ScrumMaster or coach. On many successful projects, a certain amount of naturally occurring tension exists between product owner and ScrumMaster. A product owner always desires more, more, more features. The coach, by contrast, is responsible for monitoring the health of the team. If the team is being pushed too hard and is beginning to get sloppy due to fatigue, the coach pushes back against the product

owner’s desires for more. A good way to fail at agile is to eliminate this push-pull tension between the coach and product owner by following guideline 13.

**GUIDELINE 13:** *Have one person share the roles of ScrumMaster (agile coach) and product owner. In fact, have this person also be an individual contributor on the team.*

As you can see, failure at the product owner level is easily achieved through miscommunication, general ignorance of the team’s progress, and lack of education. You can compound that, if necessary, by having one person act in roles that are designed to balance each other. Following these guidelines to the letter is a great way to fail.

## Process Issues

If all else succeeds, careful misapplication of process issues can bring down almost any agile project. Jon is a terrific example of a process nightmare, and he did most of his best sabotage without even knowing he was doing it. Jon was the lead developer for a Chicago-based team developing software designed to approve or reject loan applications. In addition to being the lead developer, he was also the ScrumMaster (note how he began by embracing guideline 13, which by itself can wreak havoc). Jon and his team were new to agile and were anxious to get rid of its unneeded parts. They immediately dispensed with daily standup meetings, reasoning that since the team sat in the same general area, most conversations could be heard over the six-foot-high cubicle walls.

They also decided that having automated unit tests was unnecessary. Since theirs was a new application, there was no chance of breaking old

code, and since all new code would be fresh in everyone’s minds there would be little chance of accidentally breaking it. However, Jon and his coworkers did embrace refactoring and collective code ownership. Their new rule was that any programmer could change the code of any other programmer at any time. They soon learned that refactoring and collective code ownership can be very dangerous without the safety net of automated unit tests to make sure you aren’t breaking things while improving them. Jon and his team had unwittingly stumbled on these two guidelines for causing a team to fail at agile.

**GUIDELINE 14:** *Start customizing an agile process before you’ve done it by the book.*

**GUIDELINE 15:** *Drop and customize important agile practices before fully understanding them.*

An alternative to these guidelines is to dive into the practices without understanding why you’re doing them. As coaches, we encounter many teams who have learned a technique or been told to do something by someone and who then continued to do it even when they’d outgrown the technique (a subtle, yet effective, subterfuge). This brings to mind the story of the newlywed wife who cuts a quarter inch off both ends of every roast she cooks. When her husband asks why she’s trimming the roast that way, she has no ready answer; she does it that way because it’s the way her mother always did it. Curious as to her mother’s rationale, the wife calls her mother and asks why she taught her to cut the ends of the roast. Her mother says she only does it that way because her own mother taught her to do so. The young wife next calls her grandmother and asks why she cut a quarter inch off the end of every



# **“Rather than align pay, incentives, job titles, promotions, and recognition with agile, create incentives for individuals to undermine teamwork and shared responsibility.”**

roast. Her grandmother tells her, “Because my roasting pan was too small. The roast wouldn’t fit any other way.” We capture this as our next guideline for failing with agile.

**GUIDELINE 16:** *Slavishly follow agile practices without understanding their underlying principles.*

If you haven’t been able to implement guidelines 14 through 16 and your agile project is succeeding despite your best efforts, you can bring even a successful project to a halt simply by changing nothing. What, you say? Change nothing? Follow the example of the StatusQ team. StatusQ, assigned to build a new Web-based reservation system, got off to a good start. Team members were new to agile but did a good level of research and sent a few of their people off to become certified ScrumMasters.

The project quickly benefited from the new practices. In a month, StatusQ had a simple Web site up and running and was able to demonstrate a few key interfaces that gave its customers a lot of confidence that their vision of an accessible and powerful reservation system would work.

StatusQ never held a retrospective at the end of each sprint. The ScrumMaster didn’t push for it because he didn’t see the need. Everything was already working wonderfully well. You can encourage this behavior on your own team by complaining loudly to your ScrumMaster and team members if they try to hold a retrospective. Tell them that it’s a waste of time to sit and talk about a project that’s going well. Tell them that retrospectives only make sense when things are going wrong.

Over time, things at StatusQ started to slow down. The rate of change and the growing code base were creating maintenance problems. Changes to the

system from the customers were supposed to be a benefit to them, but the code couldn’t keep up. Code stability became so bad that the project seemed to be moving backward. Finally company management stepped in, put a halt to the changes, and finished the contract at half price.

This team had unknowingly demonstrated our next two guidelines for failing at agile.

**GUIDELINE 17:** *Don’t continually improve.*

**GUIDELINE 18:** *Don’t change the technical practices.*

Company process issues that at first seem unrelated to the project can have a negative impact on morale and motivation. Consider the case of Dave, an up-and-coming artist who worked for a mobile phone game developer. He welcomed his company’s adoption of agile, as it made a lot of sense to him. The new agile teams consisted of programmers, artists, designers, and a number of other people from numerous disciplines. Everyone relied on each other to create iterations of their game. If the artists did a good job, then the entire team looked good. Dave often dropped what he was doing to help his teammates iterate on the art to improve the game. This often came at the expense of his own work, but, for Dave, team goals came first.

Dave’s team did a great job and produced a hit title that sold many thousands of copies and earned the company substantial profits.

At the end of the year, Dave had his first performance review with the company’s lead artist. Dave was shocked to learn that his yearly bonus was small. Dave was told that he was judged by the senior artist in the company to have missed his art production goals. As it turns out, the senior artist was counting the number of iteration task cards that

Dave completed and based his judgment on that rather than on the real amount of work Dave completed.

Chagrined, Dave returned to his team vowing to make sure his task cards took priority over the needs of the team. Guideline 19 can be your backup plan for any enthusiastic agilists in your company.

**GUIDELINE 19:** *Rather than align pay, incentives, job titles, promotions, and recognition with agile, create incentives for individuals to undermine teamwork and shared responsibility.*

A tenet shared by all agile processes is that work is prioritized based on the value provided by each feature. Other factors, such as risk and knowledge creation, are considered, but the amount of value delivered remains the dominant factor. A sure way to fail with agile is to ignore this tenet and instead follow guideline 20.

**GUIDELINE 20:** *Convince yourself that you’ll be able to do all requested work, so the order of your work doesn’t matter.*

There are, of course, other ways to fail in addition to those collected here. In your effort to undermine a successful agile project, you may have already discovered some on your own. Of course, you are probably keeping quiet about them because it’s critical that the sabotage not be detected until the bridge is blown. We are confident that, through diligent application of the guidelines here or those that only you know—or both—you will be able to plot the downfall of your next agile project. **{end}**

# Ask not what you can do for IFPUG **ASK WHAT IFPUG CAN DO FOR YOU!**

Join us for the 2008  
International Software  
**Measurement & Analysis**  
Conference



Presented by the



International Function Point Users Group

**September 14 – 19, 2008**  
**Westin Arlington Gateway Hotel**

*You won't want to forget  
what you learn at this event!*

Keynote Speaker:

- **Capers Jones, CEO, Capers Jones & Associates, LLC:**

Capers is currently the president and CEO of Capers Jones & Associates LLC. He and his colleagues have collected an abundance of historical data in an effort to judge the effectiveness of software process improvement methods. His presentation will focus on these methodologies.

Featured Speaker:

- **Bill Phifer, SEI Authorized CMMI:**

Bill, an EDS Fellow, is a Software Engineering Institute (SEI) authorized Capability Maturity Model Integration (CMMI) Lead Appraiser with over 30 years in IT including 15 years in software process implementation and improvement. His presentation will discuss the role of measurement and analysis within sourcing relationships.

IFPUG Special Event:

- **The International Spy Museum & Zola Restaurant**

**Thursday, 6:30 – 9 pm**

Join us for a fascinating private tour at one of DC's newest attractions—The International Spy Museum. Following this exclusive museum tour, join us upstairs at Zola Restaurant for an evening of delicious food and drinks.



For conference details or to register, go to [www.ifpug.org](http://www.ifpug.org)





# A GALAXY OF PATTERNS

VEER IMAGES





# JOURNEYING BEYOND THE GOF 23

BY NEIL HARRISON

**W**hy is the number twenty-three so special? It certainly isn't the answer to the Ultimate Question of Life, the Universe, and Everything. And it isn't the number of things that most people can hold in their heads at one time (that would be seven plus or minus two). Nor does it hold the significance of the numbers three, seven, twelve, and forty in Western religions. But among object-oriented software developers who consider themselves somewhat knowledgeable, the number is special—almost mystical.

## Design Patterns

Of course you know what I am talking about. There are twenty-three design patterns in the universe, right? They have memorable names, such as Observer, Singleton, Abstract Factory, and Visitor [1]. Thanks to the Gang of Four (GoF), these patterns have elevated software design and programming from an obscure craft akin to magic to a somewhat less obscure craft that merely borders on alchemy.

Now don't get all worked up that I dare to question the positive impact of the GoF patterns. To be sure, they have had a major impact on object-oriented software design throughout the world. However, because they have been so successful, too many programmers think that the GoF's twenty-three design patterns are the only patterns in the world, or at least the only ones worth knowing. This notion is even reinforced in print. The otherwise excellent book *Head First Design Patterns* [2] discusses the Null Object pattern, but relegates it to "pattern honorable mention" status. It's not a full-fledged pattern? By whose standard?

I sense that you are dubious. "What is this Null Object thing?" you ask. "And why should I care? Surely it can't be as good as Strategy, Template Method, or State." Let me explain this pattern, and you will see its value. Then perhaps you will stop believing in the inviolability of the number twenty-three.

### THE NULL OBJECT PATTERN

Before I explain what the Null Object pattern is [3], let me describe the

problem it solves. "Some object behavior is executed only in the presence of some other object. If this other object is absent, the behavior is either to do nothing or to use some default value. Using explicit conditionals to check an object reference for null and then branching, however, introduces a great deal of repetition and complexity into the code" [4]. Surely you have seen such code—you are reading along and suddenly there it is—a check to see if an object (or reference or pointer) is null. If it isn't, you continue on, but if it is, you must take some exceptional action, such as error handling. At best, this makes the code more difficult to read and understand. But it's worse than that—multiple checks in different parts of the code often lead to repeated code that handles the null case. Worst of all, though, is that it is easy for the developer to forget to check for null. This is a ticking time bomb.

The solution is surprisingly easy—"Provide something for nothing: a class that conforms to the interface required of the object reference, implementing all of its methods to do nothing, or to return suitable default values. Use an instance of this class, a so-called 'null object,' when the object reference would otherwise have been null" [4].

Or Null Object may be a singleton, thus relieving the programmer of the responsibility of remembering to delete it when a real object is used. Null Object can be profitably used in factories (Abstract Factory, Factory Method); the factory returns a real object if the input conditions are valid, and a null object if the input conditions are invalid.

By now you are beginning to see the value of Null Object. It makes the code more convenient to write and easier to understand. But Null Object has benefits beyond that. To illustrate, let's consider the usual benefits of patterns.

Why are the GoF patterns so popular? It is because they help solve the design problems we encounter again and again. If we need to encapsulate requests, the Command pattern is a convenient solution. If we have objects with different interfaces that must communicate, we can turn to Adaptor or Mediator. It makes design easier and, in many cases, facilitates code enhancement later by adding new subclasses or extending the functionality of our classes. In a few cases, a pattern might even help us write correct code. For example, Builder might help ensure that all components are properly constructed before use.

The Null Object pattern goes beyond

```
class WidgetBase {
public:
    virtual void method() = 0;
};

class RealWidget: public WidgetBase {
public:
    void method() { /* do the appropriate stuff */ }
};

class NullWidget: public WidgetBase {
public:
    void method() {} /* does nothing */
};
```

Listing 1

The structure of Null Object is simple. Listing 1 shows a sample in C++. Listing 2 shows one way to use it.

Of course, there are many variations. The methods of Null Object may actually do something, such as logging an error or taking appropriate error han-

dling actions. Or Null Object may be a singleton, thus relieving the programmer of the responsibility of remembering to delete it when a real object is used. Null Object can be profitably used in factories (Abstract Factory, Factory Method); the factory returns a real object if the input conditions are valid, and a null object if the input conditions are invalid.



```
WidgetBase* wb = new NullWidget();

...
if (some condition) {
    delete wb;
    wb = new RealWidget();
}
...
// Using it
wb->method();    // No need for if (wb), etc.
```

**Listing 2**

```
public interface LampState {
    public LampState turnSwitch();
}

public class DimState implements LampState {
    public LampState turnSwitch() {
        return new MediumState();
    }
}
```

**Listing 3**

**If you limit yourself to GoF+1, you are sadly under-informed.**

**In fact, in some cases, a GoF pattern may not be the best design.**

ment store. The cashier began to ring up the sale but then stopped and said, “I’m sorry, we have to use a different cash register. This one just went down.” I looked at the offending cash register. Emblazoned on the display was a Java message: unhandled Null Pointer exception. If the programmers who wrote that code had used Null Object, it is unlikely the cash register would have cashed out.

### THE COLLECTIONS FOR STATES PATTERN

Now we’re up to twenty-four patterns! That’s good enough, isn’t it? Well, I’m glad you like the Null Object pattern, but if you limit yourself to GoF+1, you are sadly under-informed. In fact, in some cases, a GoF pattern may not be the best design. Let’s take the lowly State pattern, for example. The concept of the State pattern is simple: Represent the states of a system as a class hierarchy, with one class for each different state. You create the proper State object for the state you are in, and it encapsulates the behavior of how the system should respond to stimuli when in that state, as shown in listing 3.

But it may not be the best solution. There are many ways to implement state-driven behavior, and using objects for states can lead to unnecessary complexity and class explosions. In fact, Frank Buschmann et al. [4] give two additional state patterns: Methods for

States and Collections for States. Let’s look at the second.

We often need to handle the lifecycle of objects or operate on them collectively with respect to their current state. While we view these objects with state, the objects themselves are independent of any state model. In such cases, the object should not be required to manage its own state. For example, a garbage collector may mark objects as referenced or unreferenced. However, the objects themselves should not have to know about this state information.

Therefore, represent each state as a collection of objects. Each object in a collection has the associated state. An object changes state by moving from one collection to another. Each collection can easily and efficiently operate on all objects of that particular state.

**Put all objects on “unreferenced” list.**

**For all objects on the “unreferenced” list:**

**If object is referenced, move it to referenced list.**

**For all objects still on the “unreferenced” list:**

**Delete them (the unreferenced list is presumably short)**

This pattern helps keep the design of the objects unencumbered by state information. At the same time, it improves efficiency where operations on collections of objects in a state are needed.

### THE LEAKY BUCKET COUNTER PATTERN

Shall we stop at GoF+2? You will find many patterns are particularly applicable to specialized domains. For example, many systems must be highly reliable and available, even in the face of real world vagaries. For example, communication software must deal with occasional link errors. A checksum error in a packet is usually a transient hiccup in the link. But many such errors indicate a real problem, and (physical) corrective measures must be taken. But how do you tell the difference between a transient fault and a real problem?

It is surprisingly easy to implement a construct that differentiates between transient and real errors. You create a counter that acts as a leaky bucket [5, 6]. When the bucket is empty, you know you have a problem.

Here’s how it works: You create a counter and initialize it to a predefined value. Every time you encounter a fault, you decrement the counter. However, at regular intervals you increment the counter, up to a predefined maximum value. As long as the faults are rare, the timer will keep the bucket (the counter) full. However, if faults happen frequently, the timer can’t keep up, and the bucket empties. When the counter reaches zero, take appropriate corrective action.

The simplest Leaky Bucket Counter takes just a few lines of code to write,



```

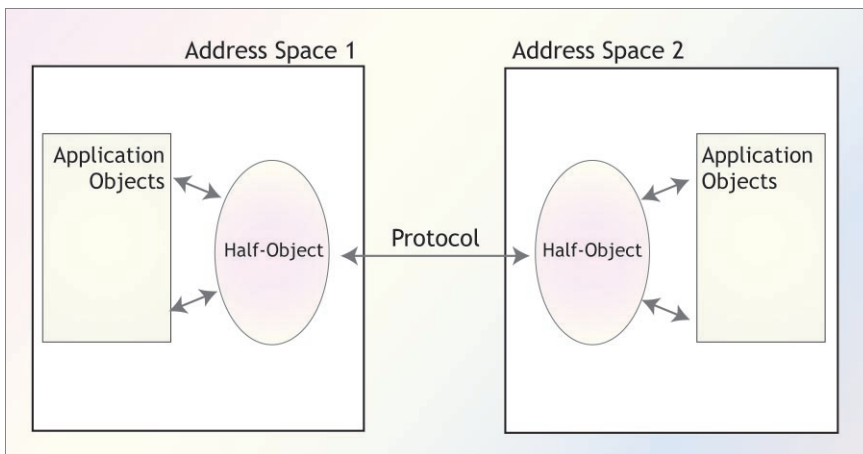
class Leaky {
public:
    Leaky (unsigned int cap) : _capacity (cap), _counter (cap) {}
    void timer_tick();
    bool got_event(); // Returns true if event threshold exceeded
private:
    const unsigned int _capacity;
    unsigned int _counter;
};

// Called from external clock function
void Leaky::timer_tick() {
    if (++_counter > _capacity)
        _counter = _capacity;
}

// Called whenever an error event happens
// Returns true if the event threshold is exceeded
bool Leaky::got_event() {
    if (--_counter > 0)
        return false;
    _counter = _capacity;           // Reset to catch next error
    return true;
}

```

**Listing 4**



**Figure 1: A protocol is required to coordinate and sync half-objects**

but requires the surrounding program to keep track of time and call the object's timer function at regular intervals, as shown in listing 4.

A more general Leaky Bucket Counter manages the time ticks internally and is only slightly more complex. However, the simple example in listing 4 illustrates the pattern.

The beauty of the Leaky Bucket Counter pattern is its simplicity. In addition, it is very low overhead, making it appropriate for high performance systems.

## THE HALF-OBJECT PLUS PROTOCOL PATTERN

The patterns I have shown you are much like the GoF patterns in that they are patterns that apply to detailed software design. There are also patterns at higher levels. For example, designers of distributed systems constantly deal with issues of performance and latency across networks when exchanging requests and responses. One way to alleviate this problem is to replicate some of the data in local address spaces, creating "half-

objects" [6, 7]. Each half-object allows some functions to be performed locally that would have been distributed. Because many operations can be done locally, the response is much faster and, overall, the necessary message traffic is reduced. A protocol is required to coordinate the half-objects and keep them in sync, as shown in figure 1.

"I can see how this improves performance," you say. "But doesn't this create a different set of problems?" Yes, it does. Data and functionality are duplicated, and there are challenges with keeping the data synchronized. All solutions to problems involve tradeoffs. Patterns describe the consequences, both positive and negative, of using particular solutions.

This is only one of many higher-level design patterns. You can use these patterns earlier in your design, before you write any code. As you implement these patterns, you often find yourself using the lower-level design patterns and language-specific idioms. Yet there are even higher-level patterns, as well. There are many architecture-level software patterns that describe the overall structure of the system, creating a view of the system shared by all developers. The best known architecture patterns include Layers, Pipes and Filters, Broker, and Model-View-Controller. In fact, I'm sure many of the systems you have worked on follow one or more of these architecture patterns. But I digress. Let's get back to patterns of design.

## Patterns and Design

Now are you learning that the pattern world does not stop at the number twenty-three? Good. Although I showed you only four additional patterns, you should understand that there are many, many others. Discover and learn them, especially those specific to the domain in which you may be working.

But there is another issue here. It may be even more important than expanding your pattern quantity horizons. You may start your work with this question, "I'm designing a system that does X. Which pattern do I use?" That's the wrong question.

You see, design patterns are not design, nor are they a substitute for design.

Notice that the proper name (the title of the GoF book notwithstanding) is patterns OF software design. I agree that this sounds like I'm splitting hairs, but it matters how we use the patterns during design.

Look at the patterns above, as well as all of the GoF patterns. They are instantiations of design decisions. You make a design decision to encapsulate the creation of objects of a class hierarchy—that's the design. Factory Method or Abstract Factory pattern is the way you make that design happen. You design your data as a hierarchal collection with recursive overtones, and Composite helps you implement the design. You design a counter to keep track of the frequency of errors and to alert you if the threshold is exceeded—that's a Leaky Bucket Counter.

What does this mean? You design your solution *following good design principles*. Along the way, patterns help you implement your design. They can even give you low level design ideas. In fact, as you use them, patterns will become a part of you, and you will draw on them more automatically in the proper circumstances.

## Design Principles

Yes, I said "following good design principles." Now you're wondering what the difference is between principles and patterns. "Doesn't that mean to design using patterns?" Sigh. That is the same attitude that fosters the belief that the design with the most patterns wins. That isn't true, but it is true that designs that follow principles of good design tend to be easier to implement and maintain. Let me illustrate the difference between software design principles and patterns.

### DON'T REPEAT YOURSELF (DRY)

This is a standard principle of good software design and coding. If you repeat code, you waste effort and leave yourself vulnerable to error propagation when forgetting to change the code the same way everywhere. A pattern that helps adhere to this principle is Null Object. Without Null Object, one is compelled to check everywhere for null references and perform some error handling (usually the same actions). Null Object con-

solidates reference checking and error handling into one place: The null object itself. The Factory patterns put all the object creation code in one place.

### LISKOV SUBSTITUTION PRINCIPLE

Barbara Liskov first intoned the advice that a derived object (subclass) should be able to be used wherever an instance of the base class (superclass) is used [8]. A derived class can always substitute for its base class. Of course, you can follow this principle without necessarily using any patterns. However, the Template Method pattern is one pattern that implements this idea.

### SEPARATE INTERFACE FROM IMPLEMENTATION

This principle allows the two to vary independently—changes to the interface need not affect the implementation, and one may change the implementation without forcing changes to the interface. One of the standard ways of doing so is to put the implementation in a class and create a separate class for the interface. Interface objects contain a reference to a corresponding implementation object. This is called variously the Handle Body Idiom (or pattern) [9] or the Pointer to Implementation idiom.

### COUPLING AND COHESION

Everyone has heard the mantra that cohesion should be high and coupling should be low. This is one of the most basic software principles, perhaps only behind modularity and information hiding. And all of these are intertwined. But here's a curious thing—not all design patterns help improve cohesion and coupling. For example, Command and Visitor allow tight coupling between objects. But, they have good reasons for doing so. And that's the point—patterns by themselves will not necessarily improve your coupling and cohesion, or any aspect of design. Instead they may help instantiate a good design.

## Putting It All Together

Well, that's a lot to think about, isn't it? I've briefly described four new patterns and hinted at many more. Yet these are just introductions to these patterns. You really need to read and study the

patterns in depth to understand the nuances of their benefits and drawbacks.

So where should you start? Ironically, don't start with the patterns. Instead, be sure to master good software design—make the principles of good design (not limited to the ones above) part of you. If you are already there, then explore patterns beyond the Gang of Four. In particular, explore patterns in the domains in which you work. Learn how these patterns help you solve problems within the overall context of disciplined software design.

Good luck with your next project. And remember that twenty-three is not the answer to the Ultimate Question of Life, the Universe, and Everything. That would be forty-two. **{end}**

---

#### REFERENCES:

- 1] Gamma, Erich; Helm, Richard; Johnson, Ralph; and Vlissides, John M. *Design Patterns: Elements of Object-Oriented Software*, Addison-Wesley, 1995.
- 2] Freeman, Eric and Freeman, Elisabeth. *Head First Design Patterns*, O'Reilly, 2004.
- 3] Woolf, Bobby. "Null Object," In *Pattern Languages of Program Design*, Volume 3, pp. 5-18. Addison-Wesley, 1998.
- 4] Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter; and Stal, Michael. *Pattern-Oriented Software Architecture*, Volume 4. Wiley, 2007.
- 5] Adams, Michael; Coplien, James; Gamoke, Robert; Hammer, Robert; Keeve, Fred; and Nicodemus, Keith. "Fault-Tolerant Telecommunication System Patterns." In *Pattern Languages of Program Design*, Volume 2, pp. 549-562. Addison-Wesley, 1995. pp. 549-562.
- 6] Utas, Greg. *Robust Communications Software*. Wiley, 2005.
- 7] Mezaros, Gerard. "Pattern: Half-Object Plus Protocol (HOPP)." In *Pattern Languages of Program Design*, Volume 1. Addison-Wesley, 1995.
- 8] Liskov, Barbara, H. and Wing, Jeanette, M. "Behavioral Subtyping Using Invariants and Constraints." Carnegie Mellon University Technical Report CMU-CS-99-156, July, 1999.
- 9] Coplien, James, O. "C++ Idioms Patterns." In *Pattern Languages of Program Design*, Volume 4. Addison-Wesley, 2000.





GIVE  
YOUR  
DEFECTS  
SOME

USING  
AUTOMATED

STATIC ANALYSIS  
TO  
DEBUG

BY GREG POPE, KIM FERRARI, AND BILL OLIVER





# ANALYZERS

# YOUR CODE

**M**ANY of this magazine's readers are familiar with the STAREAST and STARWEST conferences. Here is a trivia question for you—what does STAR stand for? Give up? It stands for Software Testing Analysis & Review. It is an acronym for the types of activities that we in the software quality field perform. However, in many organizations, the analysis activity is often overlooked, and that's too bad because analysis is a powerful tool in the quality arsenal.

Static code analysis is computer software analysis that is performed without actually executing that software. (Analysis performed on executing software is known as dynamic analysis.) In most cases, static analysis is performed on the source code. In recent years, the importance of computer security has created an expanded demand for automated tools that can analyze source code for security vulnerabilities and coding defects that could be exploited. Many security vulnerabilities are caused by questionable coding practices—for instance, using an input variable as a loop index without first checking that its value is within a valid range. Contemporary static analysis tools are able to analyze source code with a much lower false-positive rate (claiming code is defective when it is not) than previous lint-style detector tools. Because they examine only small portions of the source code at a time, lint-detector tools typically have false-positive rates of 50 percent or higher. The leading contemporary automated static analyzer (ASA) tools claim—and our experience to date has shown—false positive rates under 20 percent. These tools achieve this by parsing the source code in a way similar to compilers, creating a syntax tree and database of the entire program's code, which is then analyzed against rules or models. The ASA tools then create a report of suspected defects in the code.

Requirements	8.1%
Features and Functionality	16.2%
Structural Bugs	25.2%
Data	22.4%
Implementation and Coding	9.9%
Integration	9.0%
System Software Architecture	1.7%
Test Definition and Execution	2.8%
Other	4.7%
Sample size	6,877,000 statements (comments included)
Total defects	16,209
Bugs per 1000 statements	2.36

**Table 1**

Many of the defect types found by ASA tools would be difficult to find using peer group inspection techniques (the R in STAR). This is because a defect may be a combination of source-code statements that are physically far from each other in the source code—for instance, the allocation of memory and then, pages later, a return statement without releasing that memory. A human is limited in the amount of source code-detailed information that can be remembered from one page to the next. An ASA is not limited by this restriction, and besides, most of us do not relish the prospect of examining other people's code for hours at a time. ASA tools also can find problems that elude traditional system-level testing—for example, an array-bounds overflow, where a string of twenty characters is written into a buffer of size ten. The ten memory locations that are overwritten may not manifest a failure in the program until a later execution time or may not manifest in a repeatable way.

Since ASA tools are totally unaware of user requirements, the tools cannot replace the benefits of peer reviews or good functional testing. Also, ASA tools will not replace the need to use dynamic

analyzers. Dynamic analyzers can find problems that occur only during interactions between the executing application code, system resources, and interfaces such as race conditions.

While ASA tools are not a “silver bullet,” these tools have the capability to detect up to 50 percent of defect types and security vulnerabilities before system testing is conducted, which reduces the amount of time needed for system testing and reduces the risk of defects' escaping to the field and being discovered by customers. Table 1, from Boris Beizer's *Software Testing Techniques* [1], lists typical defect types and their percentages:

ASA tools are effective on structural bugs (which are approximately 25 percent of all bugs); 66 percent of data bugs (about 14.5 percent of all bugs); 66 percent of implementation and coding bugs (about 7 percent of all bugs); and 50 percent of integration bugs (about 4 percent of all bugs), for a total of approximately 50 percent of the types of bugs found in systems. Note that ASA tools cannot find defects related to requirements, features and functionality, and testing defects.

Because static analyzers use parsers, they are limited to specific source code

languages that they can analyze. The most common languages handled by ASA tools are C, C++, and Java. Some ASA tools can handle multiple languages because they have multiple parsers. The ASA tools with the fewest false-positive rates also compile and link the source code. Reports generated by the ASA tools assume that ASA users have a basic familiarity with the source language being analyzed. ASA tools can be used for newly developed code as well as for legacy code. They can be used as a risk-mitigation strategy for both open source code and vendor-supplied source code by incorporating ASA tools into an acceptance test procedure or release criteria. ASA tools also can be used on source code generated by automated code generators, such as code that is generated from graphical models.

As use of ASA tools becomes widespread in software quality assurance organizations, the requirement for test engineers to become familiar with the source languages used will also increase. In the early days of software development, most of the developers (who also did the testing) were of necessity familiar with the languages used. In the last three decades, specialization has evolved into developers who write code and test engineers who test the integrated code. Black box test engineers traditionally specialized in domain-specific knowledge or business requirements and were generally discouraged from understanding what the code did. The belief was that an understanding of the code (white box) would bias the thinking of the tester. Instead, the tester should have the same perspective as the customer.

ASA tools present an ideal reason for testers to upgrade their skills and learn programming languages—not so they will understand the interworking and logic of the code under test, but so they can understand the ASA tool's defect report and explain its findings to developers in a common vocabulary. Testers can use their language skills to prefilter the ASA tool's report to remove any false positives or defects that are superfluous in the context in which the code is used.

A common example of this is a call to an error-handling routine that aborts the execution of the code. Since the error

routine does not return to the calling code, an ASA-reported error of “not checking the error flag” after the error routine returns would be superfluous. Many ASA tools have the ability to be trained to ignore these types of implementation-specific errors. ASA tools can also be used by developers in the unit-testing phase of a project. Some ASA tools can be seamlessly added as a tool bar of integrated development environments (IDEs) such as Eclipse and IBM Rational Application Developer. At the time of this writing, there were seventeen open source ASA tools (primarily lint types) and thirty-eight commercial ASA tools listed on wikipedia.org.

The ASA tool used in our case studies classifies defects in four major categories:

- Defects
- Header file problems
- Low-level interface problems
- Security vulnerabilities

At the time of the case studies, the tool was capable of detecting more than fifty-two defect types from coding style, memory management, and null pointer dereferences to weak cryptography and access problems. Depending on the application, some defect types will be more critical than others. The ASA tool Klocwork provides a mechanism for tailoring the defect types detected at build time. In fact, Klocwork also provides an application-programming interface that allows an organization to create highly customized code checkers. The case studies presented here focus only on the defect checkers that are shipped with the tool and should be adequate for the vast majority of applications.

The types of defects detected are:

- Coding style
- Concurrency
- Memory-management problems
- Null pointer dereference
- Use of uninitialized data

There are eleven sub-types of defects found under coding style:

- Assignment in condition
- Inappropriate iterator usage
- Inconsistent use of types
- Invalid pointer arithmetic
- Loss of data

#### Memory Leak (mlk.must)

```

1 class A {
2     void foo();
3 };
4 void A::foo()
5 {
6     int *ptr = new int;
7     *ptr = 25;
8     ptr = new int;
9     *ptr = 35;
10 }

```

#### Klocwork produces a defect report like the following:

mlk.must.cc:8:Error:Memory leak. Dynamic memory stored in 'ptr' allocated through function 'new' at line 6 is lost at line 8  
mlk.must.cc:10:Error:Memory leak. Dynamic memory stored in 'ptr'

Figure 1

#### Memory Leak (mlk.might)

```

1 void foobar(int i)
2 {
3     char *p = (char*)malloc(12);
4     if(i) {
5         p = NULL;
6     }
7     return;
8 }

```

#### Klocwork produces a defect report like the following:

mlk.might.c:7:Error:Possible memory leak. Dynamic memory stored in 'p'

Figure 2

- Statement has no effect
- Suspicious semicolon
- Unreachable code
- Unused code
- Unused data
- Suspicious return values

There are seven sub-types of defects under memory management:

- Attempt to use memory after free
- Freeing mismatched memory
- Freeing non-heap memory
- Freeing unallocated memory
- Inconsistent freeing of memory
- Memory leak
- Returning reference to local variable

The types of header file problems detected are:

- Cycle in include files
- Missing include files
- Unnecessary include files
- Missing direct include files

The types of low-level interface problems detected are:

- Object defined in header and declared in header
- External object defined in header
- Static object defined in header
- Duplicated header
- File uses local declaration of object without using interface file
- Usage of object without declaration



#### NULL Pointer Dereference Example:

```
1 void npd_gen_must() {  
2     int *p = 0;  
3     *p = 1;  
4 }
```

#### Klocwork produces a defect report like the following:

npd.gen.must-ret-expl.c:3:3: Error(1):NPD.GEN.MUST: Null pointer 'p' that comes from line 2 will be dereferenced at line 3

Figure 3

#### Might Be a Dereference Example:

```
1 void npd_gen_might(int flag, char *arg) {  
2     char *p = arg;  
3     if (flag) p = getNull();  
4     if (arg) {p = arg;}  
5     xstrcpy(p, "Hello");  
6 }  
7  
8 void xstrcpy(char *dst, char *src) {  
9     if (!src) return;  
10    dst[0] = src[0];  
11 }
```

#### Klocwork produces a defect report like the following:

npd.gen.might-ret-call.c:5:8: Error(1):NPD.GEN.MIGHT: Null pointer 'p' that comes from call to function 'getNull' at line 3 may be dereferenced by passing argument 1 to function 'xstrcpy' at line 5.

Figure 4

#### Array Bounds Overflow Example:

```
1 int main() {  
2     char fixed_buf[10];  
3     sprintf(fixed_buf, "Very long format string\n"); return 0;  
4 }
```

#### Klocwork produces a defect report like the following:

4:Critical:Buffer overflow, array index of 'fixed\_buf' may be outside the bounds. Array 'fixed\_buf' of size 10 declared at line 3 may use index values 0..24

Figure 5

- Multiple declarations for object
- Multiple interface files
- Multi-kind object definitions
- Missing some interfaces
- Only declaration found for object
- Only definition found for object
- Global object used locally only
- Buffer overflow
- DNS spoofing
- Ignored return values
- Injection flaws
- Insecure storage
- Unvalidated user input

Access problems consist of three subtypes:

- Improper sequencing
- Least privilege
- Time of creation—time of usage

The types of security vulnerabilities detected are:

- Access problems

Insecure storage consists of two subtypes:

- Poor randomization
- Weak cryptography

Presented next are example code snippets in C/C++ and the text messages generated by Klocwork, which are examples of defects that fall into one of the four major categories. These examples are taken from the Klocwork user guide.

Figure 1 shows a memory management problem. Note in figure 1 that Klocwork has codes for each defect and has the concept of must and might. Here, memory is allocated on line 6 with the variable `ptr` and allocates memory again on line 8 with the same variable without freeing the memory first, which causes a memory leak. The method returns at line 10 without freeing the memory a second time, and since the memory was allocated within a method block, the method return goes out of scope and creates a second memory leak.

Figure 2 is an example of the might condition for the memory leak defect. Memory is allocated on line 3 and set to NULL only if the variable `i` is not zero. If `i` is zero, line 7 returns without freeing the memory.

On line 2 in figure 3, a pointer to an integer is declared and set to NULL. Then, on line 3, the pointer is dereferenced. The message at the bottom is straightforward.

In figure 4, if `arg` is NULL, a NULL pointer will be passed to the function `xstrcpy()`, which will dereference it, causing a crash. According to Klocwork's documentation, "This example illustrates that a pointer value that can come either from a local assignment of a NULL constant or from a call to a function that will return NULL, might be dereferenced either explicitly or through a call to a function that will dereference it without checking for NULL."

Notice that the message starts with Error(1):NPD.GEN.MIGHT. For each defect, Klocwork assigns a severity level. "Error" is one of those severity levels.

On line 2 of figure 5 an array of characters of size ten is declared, and on line 3 a string of size twenty-four is copied into the `fixed_buf` array, thereby overflowing the array bounds and causing

	A	B	C	D	E	F
1	src\sqlcmd.c	LOC_METHOD	FUNCTION	Lines_of_code	process_result_set	27
2	src\sqlcmd.c	NONBLANK_METHOD	FUNCTION	Non_comment_non_blank_source_lines	process_result_set	48
3	src\sqlcmd.c	NOOPUSED	FUNCTION	Number_of_operands_used	process_result_set	22
4	src\sqlcmd.c	NOOPRUSED	FUNCTION	Number_of_operators_used	process_result_set	41
5	src\sqlcmd.c	NOOPRUSED	FUNCTION	Number_of_operators_used	process_result_set	11
6	src\sqlcmd.c	NOOPRUSED	FUNCTION	Number_of_distinct_operators_used	process_result_set	2
7	src\sqlcmd.c	NOPARMS	FUNCTION	Number_of_parameters	process_result_set	7
8	src\sqlcmd.c	NOCALLS	FUNCTION	Number_of_calls	process_result_set	9
9	src\sqlcmd.c	NOCALLSOC	FUNCTION	Number_of_calls_to_methods_from_other_classes	process_result_set	16
10	src\sqlcmd.c	NOPAROTHER	FUNCTION	Number_of_parameters_passed_to_functions	process_result_set	5
11	src\sqlcmd.c	NOEXSTAT	FUNCTION	Number_of_expression_statements	process_result_set	12
12	src\sqlcmd.c	NOSTAT	FUNCTION	Number_of_statements	process_result_set	2
13	src\sqlcmd.c	NOLOPS	FUNCTION	Number_of_loops	process_result_set	2
14	src\sqlcmd.c	NOIF	FUNCTION	Number_of_conditional_statements	process_result_set	1
15	src\sqlcmd.c	NOBRANCH	FUNCTION	Number_of_branches	process_result_set	4
16	src\sqlcmd.c	MAXLEVEL	FUNCTION	Maximum_level_of_control_nesting	process_result_set	2
17	src\sqlcmd.c	AVERLEVEL	FUNCTION	Average_level_of_control_nesting	process_result_set	2

Figure 6

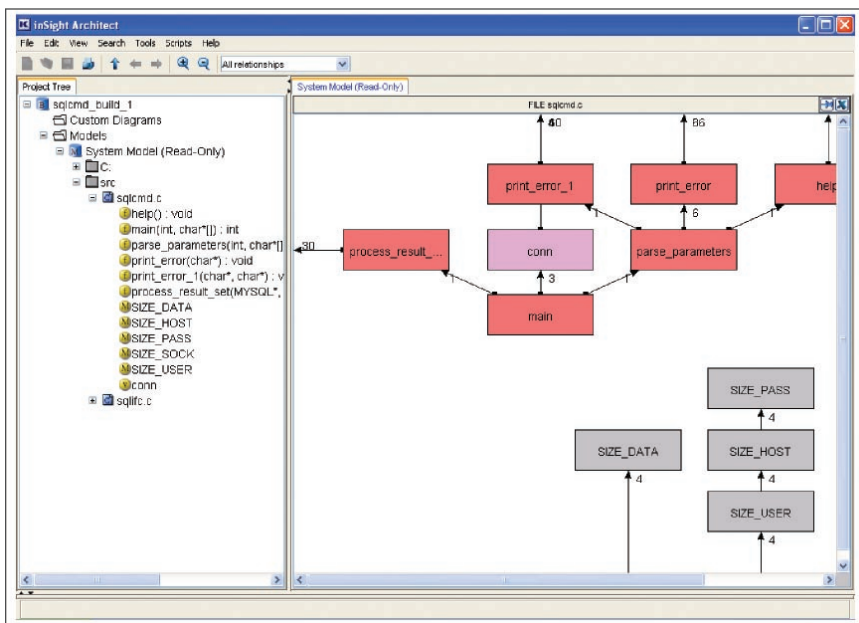


Figure 7

unpredictable and non-deterministic behavior of the program. Note the message: The severity level of this defect is "Critical," the highest level.

In addition to finding defects, the ASA tool allows users to categorize and prioritize defects found. The tool also is smart enough to recognize when defects have been fixed and that other unfixed, detected defects have shifted position in the source code. This capability is because the tool keeps track of defects by attributes and characteristics, not simply by location. The tool has a rich set of metrics, which includes cyclomatic complexity, detected fault density, and more than one hundred others, as shown in figure 6, a portion of a metrics report.

The ASA tool also contains an architectural tool that allows the source

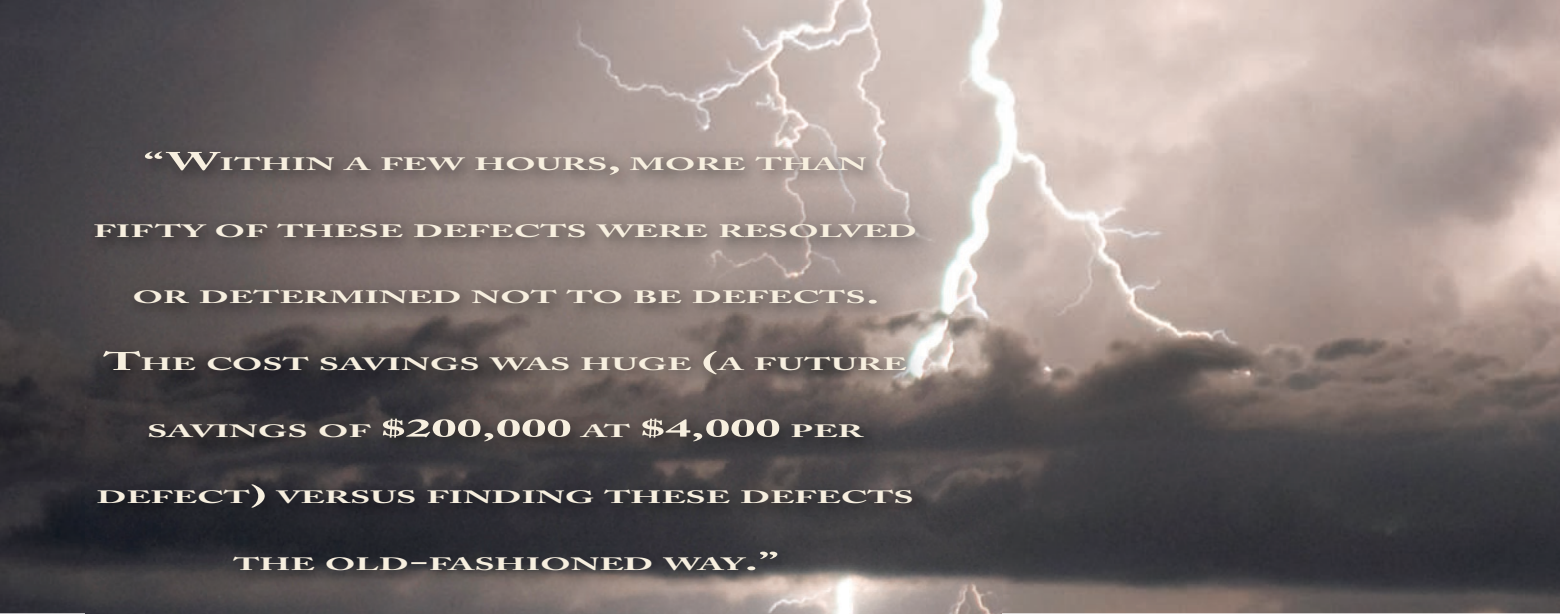
code to be displayed in graphical format. The graphs show cyclic relationships between functions or methods within files and allow the user to drag and drop the methods or functions into different files to resolve the cyclic relationships. The tool also suggests the source code changes required to re-architect the source code, as shown in the Architect tool in figure 7.

So what are some of the benefits of using ASA tools? To answer that question, consider the time it takes a developer debugging code to discover and fix software defects. Two case studies are presented. The first case study is scientific research code written in C++. This is the environment and toolkit used primarily by physicists to create highly portable 2D and 3D simulations. These pro-

grams change frequently and are highly algorithmic and complex. They also run in parallel on some of the world's largest and fastest computers. The second case study is C++ code used in security software products. This code has been stable for years with occasional functionality upgrades. These case studies represent two very different types of software.

Our first case study was done using a physics simulation application written in the C++ programming language that is used by other application development teams at Lawrence Livermore National Laboratory (LLNL), which uses two of the better performing ASA tools, Klocwork K7 and Coverity Prevent. The first case study presented is for the Klocwork K7 tool. The application in this case study involved approximately 137,000 lines of executable C++ code, and approximately 1,000 suspected defects were discovered by Klocwork. The case study focused on just one defect (buffer overflow) of which eighty-two instances were discovered. This defect type was chosen because the behavior of the code at run time is unpredictable, it isn't repeatable from run to run, it typically is not caught by debuggers, and it is one of the most time-consuming defects to isolate and repair. The value of the ASA tool became apparent when it took fewer than thirty minutes to run the static analysis on the application, and all eighty-two detected buffer overflows were fixed by the developer in just a couple of days. This translates into customers' not having to deal with intermittent and non-repeatable reliability problems and your not having to respond to as many customer complaints and release cycles with fixes, which saves money.

The second case study was conducted on 345,000 lines of code written by laboratory developers in Borland C++ Version 6. This software was used for user interfaces and gateways for a physical security system. The C++ code has been running in a 24/7 environment for four years and contains a variety of applications such as Alarm Display and Configuration Editor. This software runs at multiple sites, and the code is very stable with rare, unexplained errors. Even though the level of quality of this software was generally acceptable, our case



**“WITHIN A FEW HOURS, MORE THAN  
FIFTY OF THESE DEFECTS WERE RESOLVED  
OR DETERMINED NOT TO BE DEFECTS.  
THE COST SAVINGS WAS HUGE (A FUTURE  
SAVINGS OF \$200,000 AT \$4,000 PER  
DEFECT) VERSUS FINDING THESE DEFECTS  
THE OLD-FASHIONED WAY.”**

study attempted to determine if the tool was capable of finding the source of the rare, unexplained errors, since the consequence of these occasional errors could be large:

The ASA tool found 285 potential defects:

Critical: 56 (Null pointer dereference, buffer overflows)

Severe: 14 (Use of free memory)

Error: 193 (Memory leaks and uninitialized variables)

Warning: 21 (Inconsistent case labels)

Other: 1

only be asked to repair suspected defects that have been determined to be real. The time to prefilter—four hours for 200,000 lines of code—is still relatively small when compared to typical peer

traditional environments and mission-critical applications.

Contemporary ASA tools have proven to have three major advantages on our software projects:

- They are capable of finding about 50 percent of all possible defect types quickly, saving time and reducing development and testing costs.
- They require testers to understand computer languages.
- Prefiltering is the most time-consuming task.

Within a few hours, more than fifty of these defects were resolved or determined not to be defects. The cost savings was huge (a future savings of \$200,000 at \$4,000 per defect) versus finding these defects the old-fashioned way. It is expected that all 285 bugs could be fixed with forty hours of developer time and that traditional methods would have taken weeks if not months to fix.

Using an ASA tool requires access to the source code, external libraries, and any make and build scripts. It usually takes an hour or two to actually build the code. Prefiltering the code reports after running the build is by far the most time-consuming task. Typical prefiltering times run four hours for every 200 defects. For code with a defect rate of five per 1,000 lines, this would mean about four hours to prefilter 200,000 lines of code. Each suspected error must be categorized as either fix, not a problem, or questionable. Early experiences in handing back code directly to developers before prefiltering were unsatisfactory. Prefiltering assures that a developer will

inspection times of an hour or so to inspect 300 lines of code.

LLNL has analyzed more than 1.9 million lines of C, C++, and Java source code using Klocwork's K7. To date, more than 3,000 defects or security vulnerabilities have been discovered and repaired. It is costing about \$100 a defect to find defects using an ASA tool, and it is estimated that a defect would cost \$4,000 if found by users after release. The bottom line is that ASA technology in the higher end tools such as Coverity Prevent and Klocwork K7 works, and the return on investment easily justifies the tool. The only downside is that some prefiltering time is necessary to eliminate the 15 to 20 percent false-positive rate, the skill set of tool users requires an understanding of the source language being analyzed, and not all of the defects found are of the show stopper variety. ASA tools support the contemporary agile approaches to software development, as these tools can be run many times in a thirty-day sprint or timebox. They also can be used in

Our future plans include insertion of static-analysis capability early in the development lifecycle during the code and debugging phases by integrating the tool into the developer's IDE. Additional research is being conducted at LLNL on how to fix the code automatically after a defect is discovered. Because developers can see an immediate benefit of ASA tools, acceptance has been rapid. In fact, at LLNL, software quality engineering organizations are finding that developers actually are seeking out the tool once they understand what it can do for them. An ASA tool can have no better endorsement than that. **{end}**

LLNL-JRNL-402003-DRAFT

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

#### REFERENCES:

- 1] Boris Beizer, *Software Testing Techniques*, Second Edition, Van Nostrand Reinhold, page 57, Table 2.1.



### Virtual Lab

SEATTLE, WA—Skytap, a virtualization company formerly known as illumita, revealed its vision for the next generation of software as a service (SaaS) solutions that harness the power of virtualized infrastructure. Customers accepted into the program will receive full access to Skytap Virtual Lab capabilities, including:

- *Virtual infrastructure on demand:* Virtually unlimited hardware, software, and storage available on demand from any location and any browser. Skytap Virtual Lab scales up and down with software project demands and requires no upfront investment.
- *Automated set up and tear down of environments:* A full-featured, Web-based virtual lab automation application that eliminates manual set up and tear down tasks and enables the rapid provisioning and replication of multi-machine production environments for development and testing.

- *Skytap Library:* A pre-populated software library that includes major operating systems, databases, and applications in multiple languages, which dramatically reduces media installation tasks and enables construction of lab environments by dragging and dropping preconfigured virtual machines.
- *Collaboration in a virtual environment:* The capability to instantly collaborate on software issues and defects in a virtualized environment. Entire multi-machine lab environments can be suspended and shared with distributed, global team members to enable reproduction and diagnosis of software bugs and issues.

Visit [www.skytap.com](http://www.skytap.com) for more information.

### DevSuite 7.0

LAFAYETTE, CA—TechExcel, Inc. offers DevSuite 7.0, which enables your team to seamlessly communicate and

adapt to changes quickly, while still having enough process to give managers the control they require.

DevSuite 7.0 is the industry's first fully integrated ALM implementation platform capable of supporting agile development for projects and companies of all sizes—even distributed teams.

With DevSuite 7.0, we introduce DevSpec, a new product for requirements management. DevSpec is a full-featured offering, which can quantify requirements and software design with specifications. This enables requirements and specifications to truly drive project planning, implementation, and testing.

Visit [www.techexcel.com](http://www.techexcel.com) for more information.

### Silk 2008

AUSTIN, TX—Borland Software Corporation announces the availability of its Borland Silk 2008 product line. This new release protects customer investments in quality lifecycle products, provides a leading-edge testing framework to ensure the quality of projects while enabling broad technology support, and

S U C C E E D I N G   W I T H   A G I L E <sup>SM</sup>

## DON'T BE SHY

about transitioning to agile.

Face and overcome the challenges. Master agile software development techniques and deliver valuable, functional software in a world of uncertainty and change.

To learn more, visit  
[www.mountaingoatsoftware.com/better](http://www.mountaingoatsoftware.com/better)

Join **Mike Cohn**, author of *User Stories Applied* and *Agile Estimating and Planning*, in courses exploring the principles of agile software development: people over processes, working software over documentation, collaboration over negotiation, and flexibility over rigidity.

### Upcoming Courses

#### La Jolla

July 29-30

Certified ScrumMaster

July 31

Agile Estimating and Planning

#### San Jose

October 13

Effective User Stories

October 14-15

Certified ScrumMaster

October 16

Agile Estimating and Planning



improves enterprise usability.

Product line enhancements include:

- **Borland SilkTest 2008:** A functional and regression-testing product, SilkTest introduces a next generation “Open Agent” to deliver the latest support for Web 2.0 applications built on Adobe Flex and Microsoft Windows. Java language scripting and an Eclipse interface will be available later this quarter.
- **Borland SilkPerformer 2008:** An enterprise-class load and performance testing solution, SilkPerformer now provides broader support for customers scaling up Web 2.0 applications with testing for Adobe Flex via AMF3, and enhanced AJAX support that delivers clear readability and recognition of XML and JSON requests. Developed with input from customer BMC Software, SilkPerformer now includes comprehensive support for the Remedy Action Request System.

- **Borland SilkCentral Test Manager 2008:** The latest release of SilkCentral Test Manager exhibits customer-driven enhancements to assure reuse and management of large sets of test assets, greater flexibility for manual testers, security improvements, and significant performance and scalability updates to meet the needs of large, geographically distributed teams.

Visit [www.borland.com](http://www.borland.com) for additional information.

### DEFENSICS 3.0

SAN JOSE, CA—Codenomicon, Ltd. announces DEFENSICS 3.0, the third generation of its security and quality testing platform that allows networked product and service manufacturers and vendors to quickly identify and fix flaws by catching the problems before their offerings ever reach the market.

DEFENSICS can be installed on any industry standard workstation com-

puter, enabling customers to build whatever test environment they want. The tools can be run on carrier-grade, 64-bit multi-processor hardware with multiple 10G network interfaces, resulting in unparalleled processing power, or they can be run on a tablet PC, enabling mobility of the test system.

The DEFENSICS platform can test anything that can be linked to the security test computers through any type of wireless or wired connection and works with more than 140 wireless and wireline network protocols and connections. The platform's test tools come with pre-built test cases, which can be modified and extended. These features, combined with Codenomicon's flexible software-based platform, make the tests easy to integrate and maintain in any test environment.

Visit [www.codenomicon.com/d3/](http://www.codenomicon.com/d3/) for more information.

### Mingle 2.0

CHICAGO, IL—ThoughtWorks, Inc. announces the release of Mingle 2.0, a



## Try our self-paced eLearning course delivered right to your desktop.

Travel from your desktop with Software Quality Engineering's eLearning course, eMastering Test Design. Experience classroom value with the convenience of self-paced instruction on the Web. Because people learn in different ways and everyone learns best with multiple learning options, we've combined audio, video, text, graphics, examples, questions, exercises, and additional learning resources for the best Web-based delivery possible.

## eMastering Test Design

### Classroom Value with the Convenience of Self-paced Instruction

- Instructed and designed by two of our most experienced instructors, Lee Copeland and Rex Black
- Same valuable information as the 2-day classroom course

### The Perfect Solution for Test Practitioners with Travel and Time Constraints

- Complete an eLearning course from your own desktop
- Expert mentors provide answers to your specific questions

### Course Content

- Superior lesson material developed and delivered by testing experts
- Tutorials that place content into real-world situations
- Exercises that immediately apply your new learning
- Assessment questions that help you evaluate your learning
- Questions linked to content to reinforce your learning
- Video and audio clips enhance your learning experience
- Web access to an extensive list of additional resources
- Hyperlinks to a glossary of terms used in the course

TAKE A FREE DEMO TODAY! VISIT [WWW.SQE.COM/EMTD](http://WWW.SQE.COM/EMTD) FOR MORE INFORMATION.

major upgrade to the company's project collaboration and management software application. New features in Mingle 2.0 include:

- Cross-project reporting and one-click visibility at any level of detail—program, project, requirement, or even team member
- The ability to calculate key metrics across projects and ensure adherence to compliance requirements
- Application Program Interfaces (APIs) and plug-ins to easily integrate Mingle with existing project infrastructure

Visit [www.thoughtworks.com/mingle](http://www.thoughtworks.com/mingle) for a free thirty-day trial.

### Manual Testing

CHICAGO, IL—Original Software announces the latest version of TestDrive-Assist, its unique solution aimed at streamlining the manual testing process without taking away any control from

the tester. TestDrive-Assist is designed to help business analysts, QA teams, and user-acceptance testers in detecting and reproducing errors, sharing knowledge, and producing audit-quality diagnostics and reporting. With less than a ten-minute learning curve, users are up and running instantly, and workload is reduced by more than a third. The latest version includes a new link validation panel and enhancements to the spell checking functionality as well as the option to import a test plan to use as a checklist.

To take a test drive visit [www.manualtesting.com](http://www.manualtesting.com).

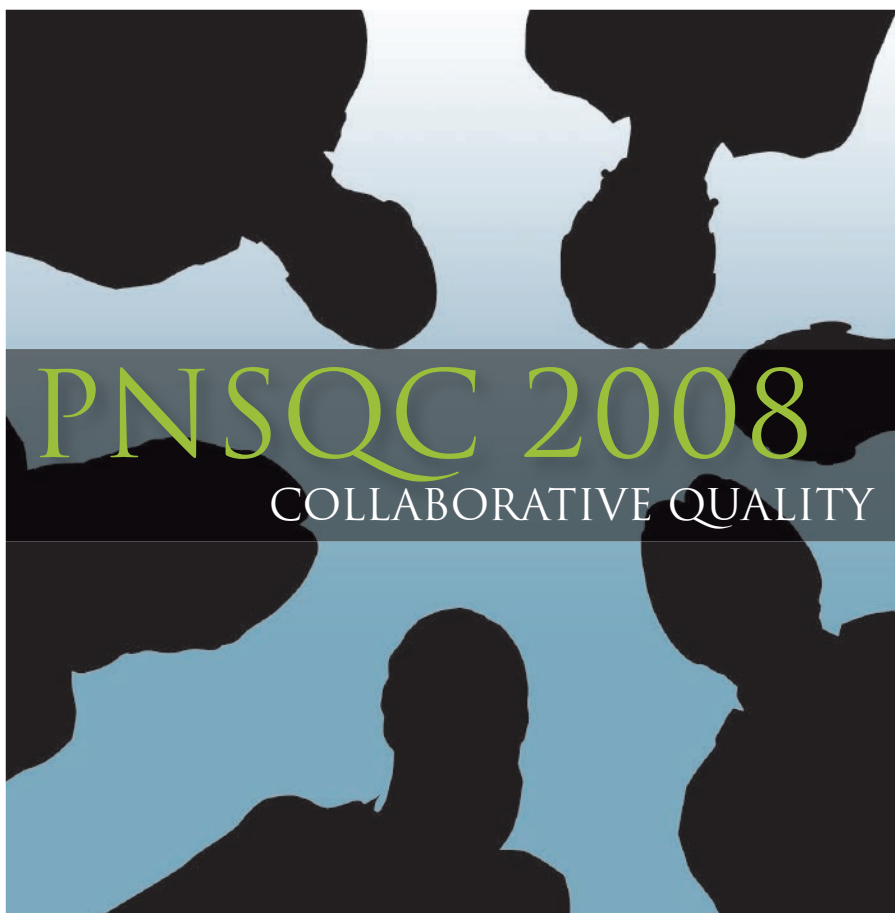


### Test & Release with Confidence

- On-demand testing via a global community of professional testers.
- Test your technology across multiple platforms, locations and environments.
- Immediate, real-world feedback to accelerate time-to-market.
- Release bug-free software with confidence.

The robust uTest testing platform is entirely Web-based — accessible anywhere, anytime, from any computer screen. The interface is easy to use and allows companies to self-manage QA cycles, testing projects and processes.

Contact us: [www.uTest.com](http://www.uTest.com) | [Sales@utest.com](mailto:Sales@utest.com) | Toll Free: 1-800-445-3914



## 26TH ANNUAL PACIFIC NORTHWEST SOFTWARE QUALITY CONFERENCE

October 13-15,  
2008  
Portland, OR

Collaboration spans levels, disciplines, and industries. Listen to these speakers discuss collaboration and how it can enable higher software quality.

- Ron Jeffries & Chet Hendrickson
- Sam Kaner
- Janet Gregory
- Karen N. Johnson
- Mike Kelly
- Steve Smith
- Tamara Suliaman & Hubert Smits

REGISTER EARLY AND SAVE	AFTER 9/8
3-Day Conference	<b>\$950</b> <b>\$1100</b>
2-Day Technical Program	<b>\$625</b> <b>\$750</b>
1-Day Workshop	<b>\$475</b> <b>\$550</b>

Special discounts available.

Visit Our New Website — [pnsqc.org](http://pnsqc.org)



# Things You Might Not Know About

## "Powering-Up" Your Existing Tests

by Dawn Haynes

Testers often come to my class looking for ways to make their testing more effective and valuable without having to make wholesale changes to their existing testing strategy. Over the years, I've collected ten heuristics for "powering up" tests by intelligently varying the conditions, sequence, data, or perspective of the tests during execution.

1

**DÉJÀ vu.** Starting from a known clean state or initial startup state, execute a test or scenario. Then, immediately repeat the same test, using the same data if possible, without cleaning up or restarting between test executions. This can expose issues with default values or states by causing the application to exhibit a different behavior or output due to subtle variables.

2

**LATHER, RINSE, REPEAT . . . AND REPEAT, AND REPEAT.** Repetition often mimics the real usage model of the application or software under test. Most users do several cycles of a small number of operations many times a day. Not testing realistic usage patterns can lead to unwanted surprises in production (like memory leaks).

3

**THE ROAD LESS TRAVELED.** Don't always follow the instructions, procedure, steps, or obvious path—many of your users won't. Some of the most costly defects in production that I have seen were found but dismissed as "training issues" or left unresolved with the declaration "a user would never do that." Even if your software is only being accessed by other systems, it's risky to assume that the other systems will always interact with yours using the same pattern or by following precisely the paths of the specifications or design.

4

**I'M A GENIUS.** Take the expert or super-user approach. Take shortcuts, jump around, skip steps, use command line interfaces, enter through back doors, stretch customizations, use undocumented features, or operate the interface quickly. Often systems and user interfaces are designed for the average user. Look for ways to "crank up" your usage.

5

**"DOH."** When walking through obvious, common, or critical scenarios, stop, smack yourself on the forehead (gently if you are prone to headaches), then try to go back and fix something, do something you forgot to do, edit an entry, etc. Basically, throw an interrupt and then change something you did by trying to Go back, Redo, Overwrite, Undo, or change Perspective or mode of usage midstream (GROUP). If your application has a UI, especially if it is browser-based, you can count on users doing some of these things when it's least desired.

6

**THINK BIG.** Specifically, think about big data. During the life of your software, it is likely to encounter large databases, large data sets, large files, large input field values, large transactions, and large volumes of transactions. Finding limits and constraints that are likely to be encountered in production, near term, or in the future can be extremely valuable in terms of planning and executing successful software implementations.

7

**PUZZLE PIECES.** Systems are often conceived and built in pieces that are later assembled in a manner intended to provide value to businesses, users, or customers. How do the pieces fit when the puzzle is completed? Think about stringing pieces together in ways that may or may not have been intended through design or workflow.

8

**BLENDER: MIX, PULSE, FRAPPÉ, CRUSH.** Blend variables, features, transactions, and usage scenarios to various degrees (mix gently vs. chaos).

9

**VARY SPACE AND TIME CONTINUUMS.** Operate the system very slowly (novice user, one-finger typist, Curious George, Sunday driver) or very fast (Speedy Gonzalez). Each of these taxes the system in a different way. Shift between fast and slow modes. Interrupt modes with things like the "coffee break test"—abandon an operation mid-stream without saving or exiting. When traveling through wormholes to get back and forth between the slow and fast universes, interesting things can happen.

10

**NATURAL DISASTERS.** Consider a few scenarios related to usage as opposed to features, functionality, and "proper" operation. Try the "shoe on the keyboard test": Put a shoe (or hand or coffee mug) on a keyboard (preferably on the ENTER key) and observe what happens. Of course, the "spilling coffee on the keyboard" and "ripping out the power cord" tests might be a bit more costly than small unexpected events. Select these wisely.

# Encourage Pair Programming

by Rob Myers

After a decade in the public eye, pair programming is still one of the most controversial of all agile practices. Appropriately, managers are concerned about the costs and developers are concerned about personal agony. I would like to enumerate some of the subtle—yet high-dividend—benefits of this discipline.

## What Is Pair Programming?

Briefly, pair programming is two people working together at one computer on one programming task. One person is the “driver” (with keyboard and mouse) and the other is the “navigator.” I’ll acknowledge that this leaves out a large number of details (e.g., physical arrangement of furniture and monitor, how often we switch roles, how often we switch people, matters of etiquette, and so on), but this definition will work for the scope of this article.

## Is It Wasteful?

In my experience, the teams that are building the most valuable software the fastest are combining the best ideas and practices from Scrum, Extreme Programming, and lean, and everything they try, they try wholeheartedly. They experience various costs; some are additional, some were hidden by other approaches. But they also experience great gains that outweigh the costs.

Though it may appear to be “two doing the work of one,” pair programming is best seen as essential collaboration. When we look at pairing from a lean perspective, we see that the benefits to the whole value stream outweigh the costs. Discouraging pair programming is a “local optimization” that prevents us from “optimizing the whole.”

## So, What Are the Benefits?

*Valuable Breaks:* Pairing encourages brief, useful breaks. Since pairs negotiate timeboxed breaks, people are less likely to meander into longer break-related

activities. Each person knows that the other is waiting.

Since break boundaries are clear, people will use the break to its greatest benefit. A short, restful break often is all that is needed to “step away” from a tough engineering problem and can lead to spontaneous breakthroughs.

*Fewer Interruptions:* People prefer not to interrupt a pair. Programmers function brilliantly when they are “in the zone,” and interruptions cause them to snap out of it for up to half an hour. This can get costly.

If you decide that it’s worth the time to interrupt, you will likely find that only one of the pair is needed to provide you with the assistance or information you require. Only one person leaves “the zone.” The other person is listening but is still capable of rapidly bringing the pair back on task.

*Fewer Defects:* It’s like a continuous code review. Pair programming would find most of the same defects as a thorough code review, but pairing is far easier to schedule and finds bugs before they set up residence in your code repository. The longer a bug resides in your code, the greater the chance that another developer—or a customer!—will start to rely on the bug’s existence.

In practice, a bug is much easier to spot by a pair of programmers at the time it’s written, because the navigator keeps the larger design context in mind and the driver is freed up to keep more of the detailed context in mind. Those mental states are difficult to achieve even a day later in the code review meeting.

*Maintainable Code:* Code is more readable. Code written by one programmer is readable by (i.e., makes

“One great way to enhance your understanding of a topic is to teach someone else from your experience.”

sense to) that one programmer at that point in time. Many of us have been forced to look at code that we wrote mere months earlier, and we scratch our heads in confusion.

Code written by two is readable by—and makes sense to—many. Understandable code leads to better maintainability and changeability. The software is more easily extended later for better profitability.

*Team-building:* Pair programming fosters a collaborative atmosphere. Real teamwork gets more work done, and people naturally stop viewing their teammates as competitors for scarce resources (e.g., bonus money). The natural competitive spirit becomes externally focused, and the team knows that it has a key role to play in beating the organization’s competition. There is less stress because there is less energy wasted figuring out how to maneuver politically within the organization.

*Cross-Training:* Intellectual cross-pollination occurs bidirectionally. A good amount of interesting and useful training occurs on the job, without any loss of productivity. Much of that training is about the overall design and architecture of the product.

New teammates assimilate culture and become productive immediately. The team can be scaled, gradually, and with less disruption. The new hire notices her own contributions immediately.

Seasoned teammates learn a lot, too, by working with people who have different backgrounds and technical experiences, and by confidently exploring together the latest technologies and how they may fit into the product.

One great way to enhance your understanding of a topic is to teach someone else from your experience. This continuous learning provides the team

with a defense against boredom, and thus attrition. You also can rely on your teammates to work well on all parts of the product, so you can finally go on vacation!

*Courage: Working with someone provides courage and tenacity.* Collaborators provide each other with sanity checks and moral support. Technical exploration tends to be fearless, focused, and immediate.

Individuals often set aside tough problems for “later”—or even for “someone else”—even if those tough tasks are at the top of the prioritized list. The pair tends to have courage and humility that an individual won’t have. For example, if I don’t know the solution to a problem, I may feel stupid. But if neither of us knows, we know that it’s a tough question. We also feel more confident asking others for help. The first thing out of my mouth doesn’t have to be “I can’t ...” but instead could be “We can’t ...” It makes a world of difference. Pairs tend to push forward or

provide feedback earlier so the business can adapt to real difficulties.

*Self-Management: Pair Programmers encourage each other to stay focused.* Whether intentionally or accidentally, workers typically burn up a lot of their day responding to email, reading news on the Web, daydreaming of an exciting job, or wasting time with other interruptions. When they pair, none of these things happen. Rather than cutting corners, they have more time in the day to do things right, thus leading to higher quality projects and fewer costly surprises. Work is both social and productive, leading to high morale and the retention of valuable employees. As a result, managers can focus their efforts on the project, the product, and the team rather than on micro-managing individuals.

## How to Adopt Pair Programming

Problems should always be examined for the root cause. This includes prob-

lems with pairing and the problems that pairing is intended to solve. Only then can the team evaluate costs and benefits honestly, with an eye toward maximizing the quality and throughput of business value for the organization. Otherwise this peculiarly effective practice may be set aside for the wrong reasons.

Rather than accepting my experience with the cost-to-benefit ratio of pair programming, you should try it yourself. Pair programming is a practice that was formalized by the agile movement. With that in mind, I suggest that the team should adopt pairing wholeheartedly for one or two iterations. Stick to your team’s pair programming “rules of engagement” diligently for that time, and then talk about it during the retrospective. Adjust accordingly. {end}

## Index to Advertisers

Agile Development Practices 2008	<a href="http://www.sqe.com/agiledevpractices">www.sqe.com/agiledevpractices</a>	8
AutomatedQA	<a href="http://www.testcomplete.com/try">www.testcomplete.com/try</a>	Inside Back Cover
Blackbaud, Inc.	<a href="http://www.blackbaud.com">www.blackbaud.com</a>	14
Eurostar	<a href="http://www.qualtechconferences.com">www.qualtechconferences.com</a>	19
Hewlett-Packard	<a href="http://www.hp.com/go/software">www.hp.com/go/software</a>	Back Cover
IFPUG	<a href="http://www.ifpug.org">www.ifpug.org</a>	29
IBM	<a href="http://www.ibm.com/rational">www.ibm.com/rational</a>	10
Klocwork	<a href="http://www.klocwork.com">www.klocwork.com</a>	Inside Front Cover
Mountain Goat Software	<a href="http://www.mountaingoatsoftware.com/better">www.mountaingoatsoftware.com/better</a>	43
PNSQC	<a href="http://www.pnsqc.org">www.pnsqc.org</a>	45
Rally Software	<a href="http://www.rallydev.com/bsm">www.rallydev.com/bsm</a>	21
STARWEST 2008	<a href="http://www.sqe.com/STARWEST">www.sqe.com/STARWEST</a>	2
SQE Agile Training	<a href="http://www.sqetraining.com/Agile">www.sqetraining.com/Agile</a>	15
SQE eLearning Training	<a href="http://www.sqetraining.com/eLearning">www.sqetraining.com/eLearning</a>	44
TechExcel	<a href="http://www.techexcel.com">www.techexcel.com</a>	1
uTest	<a href="http://www.utest.com">www.utest.com</a>	45
Web Performance, Inc.	<a href="http://www.webperformance.com">www.webperformance.com</a>	5

### Display Advertising

Shae Young [syoung@sqe.com](mailto:syoung@sqe.com)

### All Other Inquiries

[info@bettersoftware.com](mailto:info@bettersoftware.com)

*Better Software* (USPS: 019-578, ISSN: 1532-3579) is published ten times per year. Subscription rate is US \$49 per year. A US \$35 shipping charge is incurred for all non-US addresses. Payments to Software Quality Engineering must be made in US funds drawn from a US bank. For more information, contact [info@bettersoftware.com](mailto:info@bettersoftware.com) or call (800) 450-7854. Back issues may be purchased for \$15 per issue (plus shipping). Volume discounts available. Entire contents © 2008 by Software Quality Engineering (330 Corporate Way, Suite 300, Orange Park, FL 32073), unless otherwise noted on specific articles. The opinions expressed within the articles and contents herein do not necessarily express those of the publisher (Software Quality Engineering). All rights reserved. No material in this publication may be reproduced in any form without permission. Reprints of individual articles available. Call for details. Periodicals Postage paid in Orange Park, FL, and other mailing offices. POSTMASTER: Send address changes to Better Software, 330 Corporate Way, Suite 300, Orange Park, FL 32073, [info@bettersoftware.com](mailto:info@bettersoftware.com).



# Checking Automated Testing Prices?



## TestComplete™

Sensible price. Superior automated testing.

Get superior automated testing software for a fraction of what the big guys want to charge you. TestComplete is an award-winning testing solution that gives you functional, regression, unit, load, client server testing and more in one low priced package. It tests any of your Windows applications: Desktop or Web, C++ or VB, .NET or Java, Flash or WPF. Don't throw your money away on overpriced software that does less than TestComplete. **Download the free trial of TestComplete now!**

- ☒ Award-Winning Features
- ☒ Easy Record & Playback
- ☒ Great Object Recognition
- ☒ Unlimited Extensibility



FREE TRIAL - DOWNLOAD NOW  
[www.testcomplete.com/try](http://www.testcomplete.com/try)

AutomatedQA  
test, debug, deliver!  
(978) 236-7900





ALTERNATIVE THINKING ABOUT APPLICATION SECURITY:

## Hone Your Threat Detection (To A Telepathic Level).

Alternative thinking is attacking your own Web applications, finding vulnerabilities and destroying them with precision and vengeance—throughout the life of the application.

It's looking at application security through the eyes of a hacker to identify threats to your system and risks to your business.

It's harnessing the power of SPI Dynamics, recently acquired by HP, to redefine and expand your security abilities. (Please note: positive effects on your bottom line.)

It's assessing security the right way, from development to QA to operations—without slowing down the business.  
(Cue elated cheers.)

Technology for better business outcomes. [hp.com/go/securitysoftware](http://hp.com/go/securitysoftware)





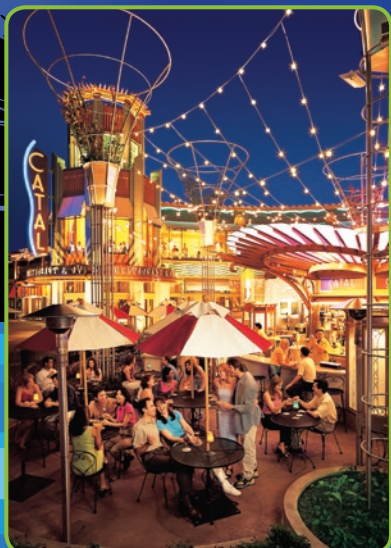
**STAR  
WEST**

# SOFTWARE TESTING

ANALYSIS & REVIEW

*The Greatest Software  
Testing Conference on Earth*

Anaheim,  
**California**



Conference  
Sponsor:

**twist**<sup>TM</sup>  
collaborative test automation

[www.sqe.com/STARWEST](http://www.sqe.com/STARWEST)

**Register Early and Save \$200!**

**September 29–October 3, 2008**

**Disneyland® Hotel**

**Over 98% of 2007 Attendees  
Recommend STARWEST  
to Others in the Industry**



As to Disney photos, logos, properties: ©Disney

## KEYNOTES



**Testing Lessons from Springfield—Home of the Simpsons**

*Rob Sabourin, AmiBug.com*



**Telling Your Exploratory Story**

*Jon Bach, Quardev, Inc.*



**Six Thinking Hats for Software Testers**

*Julian Harty, Google*



**Branch Out Using Classification Trees**

*Julie Gardiner, Grove Consultants*



**Has the Time for the Adversarial Organization Passed?**

*Gerard Meszaros, Independent Consultant*



**Testing Microsoft Office®: Experiences You Can Leverage to Drive Quality Upstream**

*Tara Roth, Microsoft*





# STAR WEST

# SOFTWARE TESTING

## ANALYSIS & REVIEW

September 29–October 3, 2008  
Disneyland® Hotel

Anaheim,  
**California**



## CONTENTS

- 
- 4 Conference-at-a-Glance  
*Build Your Own Conference!*
  - 6 Bonus Sessions and Special Events
  - 6 Software Testing Certification Training
  - 7 Conference Sponsors and Exhibitors
  - 8 35 In-Depth Pre-conference Tutorials
  - 16 6 Keynote Presentations
  - 18 45 Concurrent Sessions
  - 27 Registration Information
  - 27 Event Location
  - 27 Ways to Save

## THE TOP TEN REASONS TO ATTEND STARWEST

1. Over 100 learning sessions: tutorials, keynotes, conference sessions, bonus sessions, and more
2. In-depth tutorials, half- and full-day options—double the number of classes from last year
3. Cutting-edge testing answers from top testing experts
4. Presentations from highly experienced testing professionals
5. Opportunities to network with your peers in the industry
6. Special events—welcome reception, book signings, Meet the Speakers, and more
7. The largest testing EXPO anywhere
8. Group discounts—bring your whole team
9. The perfect balance of learning and fun in Southern California
10. All this at the happiest place on Earth—*Disneyland®!*

## WHO'S BEHIND THE CONFERENCE?



Software Quality Engineering assists professionals interested in improving software practices. Four conferences are hosted annually—the STAR conference series, the Better Software Conference & EXPO, and Agile Development Practices. Software Quality Engineering also delivers software training, publications, and research. [www.sqe.com](http://www.sqe.com)



Better Software magazine brings you the hands-on facts you need to run smarter projects and to deliver better products that win in the marketplace. [www.BetterSoftware.com](http://www.BetterSoftware.com)



StickyMinds.com is a complete online resource to help you produce better software. It offers original articles from industry experts, technical papers, industry news, a tools guide, forums, and much more. [www.StickyMinds.com](http://www.StickyMinds.com)

# THE TESTING EXPO

October 1–2, 2008

## Visit Top Industry Providers Offering the Latest in Testing Solutions

Looking for answers? Take time to explore this one-of-a-kind EXPO, designed to bring you the latest solutions in testing technologies, software, and tools. To support your software testing efforts, participate in technical presentations and demonstrations conducted throughout the EXPO.

Meet one-on-one with representatives from some of today's most progressive and innovative organizations.



## EXPO Hours

### Wednesday, October 1

10:30 a.m. – 2:00 p.m.

3:30 p.m. – 6:30 p.m.

Reception: 5:30 p.m. – 6:30 p.m.

*All attendees are invited to the EXPO reception for complimentary food and beverages.*

### Thursday, October 2

10:30 a.m. – 3:00 p.m.

For Sponsor/Exhibitor news and updates, visit [www.sqe.com/STARWEST](http://www.sqe.com/STARWEST).

See page 7 for a preview of Sponsors and Exhibitors.

## NETWORKING WITH COLLEAGUES

*You asked for it . . . and we deliver. Experience more ways to network with peers, across all industries, at STARWEST 2008.*

- **Welcome Reception**—Kick off the conference and enjoy complimentary food and beverages with your colleagues
- **The Testing EXPO**—Look for answers to your testing needs and meet other attendees with the same challenges
- **EXPO Reception**—Socialize with others and enjoy complimentary food and beverages
- **Bonus Sessions**—Connect with both colleagues and industry experts during these new sessions to enhance your learning experience
- **Testing Dialogues: Management Issues**—Get advice from your peers and testing experts regarding your current challenges
- **Meet the Speakers**—Pose your toughest questions to industry experts
- **Speaker Book Signings**—Meet the author of your favorite book
- **StickyMinds.com Testing Challenge**—See how your testing skills rank against the skills of others
- **Breakfasts, Breaks, Lunches, and More**



## WHAT TO DO WHILE YOU'RE IN DISNEYLAND® AND DOWNTOWN DISNEY®

- Celebrate more than fifty magical years at Disneyland®
- Ride the California Screamin' roller coaster at Disney's California Adventure®
- Watch in wonder as the state-of-the-art fireworks spectacular bursts across the sky high above Sleeping Beauty's Castle
- Amble down to Main Street, U.S.A. for the larger-than-life Parade of Dreams, featuring more than fifty of your favorite Disney Characters, dazzling floats, amazing performers, and unbelievable special effects
- Enjoy music, shopping, and dining at Downtown Disney®
- Visit the House of Blues and get your fill of delicious food and live music
- Visit the ESPN Zone® restaurant and sports complex featuring non-stop action for the entire family



## EXTEND YOUR STAY AND ENJOY WHAT SOUTHERN CALIFORNIA HAS TO OFFER

- Scream on GhostRider®, Orange County's first wooden roller coaster, at Knott's Berry Farm®
- Pay tribute to the Surfing Hall of Fame at the International Surfing Museum
- Visit Mission San Juan Capistrano, one of California's oldest Spanish missions
- Take a day trip to Catalina Island—California's only island resort experience
- Take a stroll around San Diego Zoo and visit with giant pandas, koalas, and Komodo dragons
- Enjoy a nice glass of wine in Temecula, a picturesque wine country featuring world-class wineries, tasting rooms, wine country tours, dining, lodging, and year-round special events

As to Disney photos, logos, properties: ©Disney  
Photos courtesy of Anaheim/Orange County Visitor & Convention Bureau

# CONFERENCE-AT-A-GLANCE

## SUNDAY, SEPTEMBER 28

**8:30** **Software Testing Certification—Certified Tester – Foundation Level Training** (8:30 a.m. - 5:00 p.m., with lunch from 12:00 p.m. - 1:00 p.m.)

## MONDAY, SEPTEMBER 29

**8:30** Tutorial Sessions (8:30 a.m. - 12:00 p.m.)

### MONDAY FULL-DAY TUTORIALS

- MA** **Becoming an Influential Test Team Leader**  
*Randall Rice, Rice Consulting Services, Inc.*
- MB** **Introduction to Systematic Testing** – Dale Perry, Software Quality Engineering
- MC** **The Art and Science of SOA Testing**  
*Rizwan Mallal and Mamoon Yunus, Crosscheck Networks*
- MD** **Exploratory Software Testing Explained** – Jonathan Kohl, Kohl Concepts, Inc.
- ME** **NEW How to Teach Yourself Testing** – James Bach, Satisfice, Inc.
- MF** **Transition to Agile Development: A Tester's View** – Bob Hartman, Net Objectives
- MG** **NEW Scripting Techniques for Testers** – Dion Johnson, DJohn Innovative Consulting, Inc.
- MH** **Key Test Design Techniques** – Lee Copeland, Software Quality Engineering

### MONDAY MORNING TUTORIALS

- MI** **NEW Becoming a Trusted Advisor to Senior Management**  
*Lloyd Roden, Grove Consultants*
- MJ** **Reliable Test Effort Estimation** – Ruud Teunissen, POLTEQ IT Services BV
- MK** **Using Visual Models for Test Case Design** – Rob Sabourin, AmiBug.com, Inc.
- ML** **Measurement and Metrics for Test Managers**  
*Rick Craig, Software Quality Engineering*
- MM** **NEW Difficult Testing Questions and How to Answer Them**  
*Michael Bolton, DevelopSense*

**12:00** Lunch

**1:00** Tutorial Sessions (1:00 p.m. - 4:30 p.m.)

### MONDAY FULL-DAY TUTORIALS (CONTINUED)

- MA** **Becoming an Influential Test Team Leader**  
*Randall Rice, Rice Consulting Services, Inc.*
- MB** **Introduction to Systematic Testing** – Dale Perry, Software Quality Engineering
- MC** **The Art and Science of SOA Testing**  
*Rizwan Mallal and Mamoon Yunus, Crosscheck Networks*
- MD** **Exploratory Software Testing Explained** – Jonathan Kohl, Kohl Concepts, Inc.
- ME** **NEW How to Teach Yourself Testing** – James Bach, Satisfice, Inc.
- MF** **Transition to Agile Development: A Tester's View** – Bob Hartman, Net Objectives
- MG** **NEW Scripting Techniques for Testers** – Dion Johnson, DJohn Innovative Consulting, Inc.
- MH** **Key Test Design Techniques** – Lee Copeland, Software Quality Engineering

### MONDAY AFTERNOON TUTORIALS

- MN** **Risk-Based Testing** – Julie Gardiner, Grove Consultants
- MO** **Spend Wisely, Test Well: Making a Financial Case for Testing**  
*Susan Herrick, EDS-Global Testing Practice*
- MP** **To Infinity and Beyond: Extreme Boundary Testing** – Rob Sabourin, AmiBug.com
- MQ** **The Craft of Bug Investigation** – Jon Bach, Quardev, Inc.
- MR** **UPDATED Managing Test Outsourcing** – Martin Pol, POLTEQ IT Services BV

**5:00** Certification Information Session (see page 6 for details)

## TUESDAY, SEPTEMBER 30

**8:30** Tutorial Sessions (8:30 a.m. - 12:00 p.m.)

### TUESDAY FULL-DAY TUTORIALS

- TA** **Adapting to Agile** – Elisabeth Hendrickson, Quality Tree Software, Inc.
- TB** **How to Build, Support, and Add Value to Your Test Team**  
*Lloyd Roden and Julie Gardiner, Grove Consultants*
- TC** **Essential Test Management and Planning** – Rick Craig, Software Quality Engineering
- TD** **Understanding Software Performance Testing**  
*Dale Perry, Software Quality Engineering*
- TE** **Just-In-Time Testing** – Rob Sabourin, AmiBug.com, Inc.
- TF** **Requirements-Based Testing** – Richard Bender, Bender RBT, Inc.
- TG** **Session-Based Exploratory Testing** – Jon Bach, Quardev, Inc.
- TH** **UPDATED Test Process Improvement**  
*Martin Pol and Ruud Teunissen, POLTEQ IT Services BV*
- TI** **NEW Protecting Your Applications from Web Security Vulnerabilities**  
*Caleb Sima, Hewlett Packard*

### TUESDAY MORNING TUTORIALS

- TJ** **Test Automation Patterns: Best Practices and Common Pitfalls**  
*Gerard Meszaros, Independent Consultant*
- TK** **High Impact Software Inspections**  
*Ed Weller, Integrated Productivity Solutions, LLC*
- TL** **NEW Agile Defect Reporting With User Stories and Tests**  
*Antony Marciano, testingReflections.com*
- TM** **NEW AJAX Testing: Inside and Out** – Paco Hope, Cigital

**12:00** Lunch

**1:00** Tutorial Sessions (1:00 p.m. - 4:30 p.m.)

### TUESDAY FULL-DAY TUTORIALS (CONTINUED)

- TA** **Adapting to Agile** – Elisabeth Hendrickson, Quality Tree Software, Inc.
- TB** **How to Build, Support, and Add Value to Your Test Team**  
*Lloyd Roden and Julie Gardiner, Grove Consultants*
- TC** **Essential Test Management and Planning** – Rick Craig, Software Quality Engineering
- TD** **Understanding Software Performance Testing**  
*Dale Perry, Software Quality Engineering*
- TE** **Just-In-Time Testing** – Rob Sabourin, AmiBug.com, Inc.
- TF** **Requirements-Based Testing** – Richard Bender, Bender RBT, Inc.
- TG** **Session-Based Exploratory Testing** – Jon Bach, Quardev, Inc.
- TH** **UPDATED Test Process Improvement**  
*Martin Pol and Ruud Teunissen, POLTEQ IT Services BV*
- TI** **NEW Protecting Your Applications from Web Security Vulnerabilities**  
*Caleb Sima, Hewlett Packard*

### TUESDAY AFTERNOON TUTORIALS

- TN** **NEW Introducing Keyword-Driven Test Automation**  
*Hans Buwalda, LogiGear Corporation*
- TO** **Essential Thinking Skills for Testers and Test Managers**  
*Krishna Iyer and Mukesh Mulchandani, ZenTEST Labs*
- TP** **NEW Cover Your Testing Bases with the Agile Testing Quadrants**  
*Lisa Crispin, ePlan Services, Inc.*
- TQ** **The Fundamentals of Pairwise Testing—With a Twist**  
*Randall Rice, Rice Consulting Services, Inc.*

**4:30** **Welcome Reception—Sponsored by: Twist from ThoughtWorks** (4:30 p.m. - 6:00 p.m.)

**6:00** **BONUS SESSION—Speaking 101: Tips and Tricks** – Lee Copeland, Software Quality Engineering



## WEDNESDAY, OCTOBER 1

8:30	Opening Remarks — Conference Chair - Lee Copeland, Software Quality Engineering																													
8:45	Testing Lessons from Springfield—Home of the Simpsons — Rob Sabourin, AmiBug.com																													
10:00	Telling Your Exploratory Story — Jon Bach, Quardev, Inc.																													
11:00	Networking Break • Visit the Testing EXPO, 10:30 a.m. – 2:00 p.m.																													
11:30	<table><tr><th colspan="2">Test Management</th><th colspan="2">Test Techniques</th><th colspan="2">Test Automation</th><th colspan="2">Personal Excellence</th><th colspan="2">Special Topics</th></tr><tr><td>W 1</td><td>The Myth of Risk Management Pete McBreen, Software Craftsmanship, Inc.</td><td>W 2</td><td>Using Failure Modes to Power Up Your Testing Dawn Haynes, PerTestPlus, Inc.</td><td>W 3</td><td>Adventures with Test Monkeys John Fodeh, Hewlett-Packard</td><td>W 4</td><td>Five Things Every Tester Must Do Julie Gardiner, Grove Consultants</td><td>W 5</td><td>Fun with Regulated Testing John McConda, Mobius Test Labs</td></tr></table>										Test Management		Test Techniques		Test Automation		Personal Excellence		Special Topics		W 1	The Myth of Risk Management Pete McBreen, Software Craftsmanship, Inc.	W 2	Using Failure Modes to Power Up Your Testing Dawn Haynes, PerTestPlus, Inc.	W 3	Adventures with Test Monkeys John Fodeh, Hewlett-Packard	W 4	Five Things Every Tester Must Do Julie Gardiner, Grove Consultants	W 5	Fun with Regulated Testing John McConda, Mobius Test Labs
	Test Management		Test Techniques		Test Automation		Personal Excellence		Special Topics																					
W 1	The Myth of Risk Management Pete McBreen, Software Craftsmanship, Inc.	W 2	Using Failure Modes to Power Up Your Testing Dawn Haynes, PerTestPlus, Inc.	W 3	Adventures with Test Monkeys John Fodeh, Hewlett-Packard	W 4	Five Things Every Tester Must Do Julie Gardiner, Grove Consultants	W 5	Fun with Regulated Testing John McConda, Mobius Test Labs																					
12:30	Lunch • Visit the Testing EXPO																													
1:45	W 6	Great Test Teams Don't Just Happen Jane Fraser, Electronic Arts	W 7	Understanding Test Coverage Michael Bolton, DevelopSense	W 8	Automate API Tests with Windows PowerShell Nikhil Bhandari, Intuit	W 9	What Price Truth? When a Tester is Asked to Lie Fiona Charles, Quality Intelligence, Inc.	W 10	The Case Against Test Cases James Bach, Satisfice, Inc.																				
3:00	W 11	Test Estimation: Painful or Painless? Lloyd Roden, Grove Consultants	W 12	Exploratory Testing: The Next Generation David Gorena Elizondo, Microsoft	W 13	Test Automation Techniques for Dynamic and Data Intensive Systems Chris Condon, The Hanover Group	W 14	Top Ten Non-Technical Skills for Better Testing Krishna Iyer and Mukesh Mulchandani, ZenTEST Labs	W 15	The Power of Specially Gifted Software Testers Thorkil Sonne, Specialisterne																				
4:00	Networking Break • Visit the Testing EXPO, 3:30 p.m. – 6:30 p.m.																													
4:30	Six Thinking Hats for Software Testers — Julian Harty, Google																													
5:30	Reception in the EXPO Hall, 5:30 p.m. – 6:30 p.m.																													

## THURSDAY, OCTOBER 2

8:30	Branch Out Using Classification Trees — Julie Gardiner, Grove Consultants									
9:45	Test Management		Test Techniques		Test Automation		The New Wave		Special Topics	
	T 1	Quality Metrics for Testers: Evaluating Our Products, Evaluating Ourselves Lee Copeland, Software Quality Engineering	T 2	Patterns and Practices for Model-Based Testing Keith Stobie, Microsoft	T 3	End-to-End Test Automation for Complex Systems Thomas Thunell, Ericsson AB	T 4	Testing AJAX Applications with Open Source Tools Frank Cohen, PushToTest	T 5	Man and Machine: Combining Tools with the Human Mind Jonathan Kohl, Kohl Concepts, Inc.
	BONUS SESSION—Testing Dialogues: Management Issues — Rob Sabourin, AmiBug.com and Jon Bach, Quardev, Inc.									
10:45	Networking Break • Visit the Testing EXPO, 10:30 a.m.–3:00 p.m.									
11:15	T 6	Calculate the Value of Testing: It's Not Just About Cost Leo van der Aalst, Sogeti Netherlands BV	T 7	The Savvy Web Tester's Tool Kit Erik Petersen, emprove	T 8	Demystifying Virtual Test Lab Management Ian Knox, Skytap, Inc.	T 9	Building an SOA Quality Center of Excellence Rajeev Gupta, iTKO LISA	T 10	Beyond Functional Testing: On to Conformance and Interoperability Derk-Jan De Grood, Collis
12:15	Lunch • Visit the Testing EXPO • Meet the Speakers									
1:30	Test Management		Test Techniques		Agile Testing		Performance Testing		Special Topics	
	T 11	Managing Your Personal Stress Level Randall Rice, Rice Consulting Services, Inc.	T 12	Reloadable Test Data for Manual Testing Tanya Dumaresq, Macadamian Technologies, Inc.	T 13	Driving Development with Tests: ATDD and TDD Elisabeth Hendrickson, Quality Tree Software, Inc.	T 14	Performance Engineering: More Than Just Load Testing Rex Black, QA Software Consultant/Trainer	T 15	Test Management for Very Large Programs: A Survival Kit Graham Thomas, Independent Consultant
	Networking Break • Visit the Testing EXPO									
3:00	T 16	The Three Faces of Quality: Control, Assurance, Analysis Stephen Michaud, Luxoft Canada	T 17	Acceptable Acceptance Testing Grigori Melnik, Microsoft Corporation and Jon Bach, Quardev, Inc.	T 18	Are Agile Testers Different? Lisa Crispin, ePlan Services, Inc.	T 19	Life as a Performance Tester Scott Barber and Dawn Haynes, PerfTestPlus, Inc.	T 20	Adding Measurement to Reviews Riley Rice, Booz Allen Hamilton
4:15	Has the Time for the Adversarial Organization Passed? — Gerard Meszaros, Independent Consultant									
5:30	BONUS SESSION—State of the Software Testing Industry — Panel Discussion									
	Certification Exam (See page 6 for details)									

## FRIDAY, OCTOBER 3

8:30	Testing Microsoft Office®: Experiences You Can Leverage to Drive Quality Upstream — Tara Roth, Microsoft								
9:30	Networking Break								
10:00									
	Test Management	Test Techniques	Agile Testing	Security	Special Topics				
F 1	Toward an Exploratory Testing Culture Rob Sabourin, AmiBug.com, Inc.	F 2	Truths and Myths of Static Analysis Paul Anderson, GrammaTech	F 3	Lessons Learned in Acceptance Test-Driven Development Antony Marcano, testingReflections.com	F 4	Automating Security Testing with cUrl and Perl Paco Hope, Cigital	F 5	Database Locking: What Testers Should Know, Why Testers Should Care Justin Callison, Luxoft Canada
F 6	Keys to Successful Test Outsourcing: Win-Win for All Patricia Smith, The Hartford	F 7	A Modeling Framework for Scenario-Based Testing Fiona Charles, Quality Intelligence, Inc.	F 8	Agile Acceptance Testing Using .NET FitNesse Gojko Adzic, Neuri Ltd.	F 9	Integrating Security Testing into Your Process Danny Allan, IBM Rational	F 10	Going Mobile: The New Challenges for Testers Wayne Horn, Augmentum
11:15									
1:15	BONUS SESSION—Getting Things Done—Putting the Lessons of STARWEST 2008 to Work — Rob Sabourin, AmiBug.com								

## BONUS SESSIONS

### Speaking 101: Tips and Tricks—with Lee Copeland

**Tuesday, September 30, 6:00 p.m. - 7:00 p.m.**

Are you a new STAR speaker or aspiring to be one in the future? Join us at this workshop on making effective conference presentations. Learn the secrets of developing content, identifying the Big Message, preparing slides with just the right words and images, presenting your message, handling questions from the audience, and being ready when things go wrong. Lee Copeland, a professional speaker since birth, shares ideas that will help you be a better speaker, no matter what the occasion.

### Testing Dialogues: Management Issues—with Rob Sabourin, AmiBug.com and Jon Bach, Quardev, Inc.

**Thursday, October 2, 9:45 a.m. - 12:15 p.m.**

In your role as a test manager, where are you struggling? Would you like advice from your peers and testing experts regarding your current challenges? Testing Dialogues is a unique opportunity for you to learn from others and to share your challenges, ideas, and solutions. Facilitated by Rob Sabourin and Jon Bach, two noted testing professionals, this double-track session focuses on the issues test managers face every day. You'll shape the discussion by identifying topics of interest, joining with others to discuss these topics, and sharing and learning. Take this opportunity to structure your own conference session dealing with your real-life issues.

### State of the Software Testing Industry—Panel Discussion

**Thursday, October 2, 5:30 p.m. - 6:30 p.m.**

There's a great deal of buzz about the future of software testing and quality. It is natural to speculate about innovations, trends and blind alleys—realistically, what works and what does not. Before we can understand where we are likely to be going or where we want to be, it helps to have a realistic sense of where we are today. This facilitated bonus session will be a hearty and illuminating exchange of ideas.

### Getting Things Done—Putting the Lessons of STARWEST 2008 to Work—with Rob Sabourin

**Friday, October 3, 1:15 p.m. - 3:15 p.m.**

With so many presentations and so much to offer, STARWEST 2008 is committed to helping you put new information and techniques into practice. Stay after the conference concludes for a facilitated workshop with Rob Sabourin to help you integrate ideas and organize a plan of action. Participants' input will define the issues and agenda topics.

## SPECIAL EVENTS

### Welcome Reception

**Tuesday, September 30, 4:30 p.m. - 6:00 p.m.**

Help us kick off the STARWEST 2008 conference with a Welcome Reception and enjoy complimentary food and beverages.

### EXPO Reception

**Wednesday, October 1, 5:30 p.m. - 6:30 p.m.**

Socialize with others and enjoy complimentary food and beverages in the EXPO hall Wednesday evening. Look for answers to your testing needs and meet other attendees with the same challenges.

### Meet the Speakers

**Thursday, October 2, During Lunch**

Meet with industry experts for an open discussion in key areas of software testing technology. Pose your toughest questions, address specific project needs, gather details on the latest research and cutting-edge testing practices, or just come by to say hello.

### StickyMinds.com Testing Challenge

**Wednesday, October 2, and Thursday, October 3, In the EXPO**

StickyMinds.com invites you to try out your testing skills at the Testing Challenge during EXPO hours. How do you rank against fellow testers? Take the challenge in the EXPO!

### Bookstore and Speaker Book Signings

During EXPO hours, purchase popular industry books—many authored by STARWEST speakers—from BreakPoint Books. Authors are available for questions and book signings during session breaks and EXPO hours.

### Certification Exam and Information Session

You have the opportunity to take the ISTQB™ Certified Tester-Foundation Level exam facilitated by the American Software Testing Qualifications Board (ASTQB) at STARWEST for \$250. To pre-register for the exam or to download the syllabus, visit [www.astqb.org](http://www.astqb.org). The examination will be held on site **Thursday, October 2, at 5:30 p.m.** There will be a free information session to introduce you to the certification program and examination on **Monday, September 29, at 5:00 p.m.**



## SOFTWARE TESTING CERTIFICATION TRAINING AT STARWEST

### Software Testing Certification Training Certified Tester—Foundation Level

**Sunday, Sept. 28 – Tuesday, Sept. 30  
8:30 a.m. – 5:00 p.m.**



### Are you looking for internationally recognized certification in software testing?

Delivered by top experts in the testing industry, Software Testing Certification-Foundation Level is an accredited training course, designed to help prepare you for the ISTQB™ Certified Tester-Foundation Level exam. This certification program, accredited by the ISTQB™ through its network of National Boards, is the only internationally accepted certification for software testing. The ISTQB™, a non-proprietary and non-profit organization, has granted more than 80,000 certifications in more than thirty-two countries around the world.

Through the Software Testing Certification-Foundation Level training course, learn the basics needed to become a software test professional and understand how testing fits into the software development. Find out what it takes to be a successful software test engineer and how testing can add significant value to software development. Study all of the basic aspects of software testing, including a comprehensive overview of tasks, methods, and techniques for effectively testing software. In addition, learn the

fundamental steps in the testing process: planning, analysis, design, implementation, evaluation, and reporting.

The Software Testing Certification—Foundation Level course covers the topics needed to prepare you for the ISTQB™ Certified Tester—Foundation Level exam:

- Fundamentals of software testing – Concepts and context, risk analysis, goals, process, and psychology
- Lifecycle testing – How testing relates to development including models, verification and validation, and types of tests
- Static testing – Reviews, inspections, and static tools
- Test design techniques – Black-box test methods, white-box techniques, error guessing, and exploratory testing
- Test management – Team organization, key roles and responsibilities, test strategy and planning, configuration management, defect classification and management
- Testing tools – Tool selection, benefits, risks, and classifications

The Software Testing Certification training program is appropriate for individuals who recently entered the testing field and those currently seeking certification in testing.

### REGISTER EARLY—SPACE IS LIMITED!

At 3:30 p.m. on the third day of the course, the ISTQB™ Certified Tester-Foundation Level exam will be given. The ISTQB™ Certified Tester-Foundation Level certification exam is independently administered by the American Software Testing Qualifications Board. A \$250 fee for the exam is included in your course registration. For more information on ISTQB™ certification or to download the syllabus, please visit [www.astqb.org](http://www.astqb.org).

### You save an additional \$200 by attending both the Certification Training and the Conference!

Register for the Certification Course and the STAR testing conference at the same time to save an extra \$200. Contact Client Support at 888.268.8770 or 904.278.0524 or [sqeinfo@sqe.com](mailto:sqeinfo@sqe.com).

## CONFERENCE SPONSOR



**Twist™** enables domain experts and testing professionals to jointly build test suites of lasting business value and impact. It is a next-generation platform with a rich environment for authoring, executing, and maintaining tests. Twist is created by ThoughtWorks, pioneers of distributed Agile development, and authors of Selenium, the world-leading web testing framework. Twist is being launched at STARWEST 2008. [www.thoughtworks.com/twist](http://www.thoughtworks.com/twist)

## INDUSTRY SPONSORS



**Cognizant** (NASDAQ: CTSI) provides IT services focused on delivering strategic solutions that address client's complex business needs. As the largest provider of Independent Verification & Validation services, Cognizant Testing Services provides comprehensive solutions including functional testing, shared services, specialized services, consulting and enterprise testing services. For additional information, visit [www.cognizant.com/testing](http://www.cognizant.com/testing)



**HP** is a technology solutions provider to consumers, businesses and institutions. Our business technology optimization (BTO) products make up the industry's most comprehensive suite of IT management software and help IT make every dollar deliver positive business outcomes. HP has paired Mercury's years of QA experience and SPI Dynamics' security expertise with the full range of HP solutions and support to lower costs, improve delivery time, and reduce risk. [www.hp.com/go/software](http://www.hp.com/go/software)



**IBM Rational Quality Management.** Software development teams continually struggle to balance schedules, resources and functionality. Rational Quality Management solutions enable our customers to implement continuous processes, powered by automation to govern the process of software delivery ensuring on time and on budget delivery of applications that clearly address business objectives. [www.ibm.com/software/rational/offering/testing.html](http://www.ibm.com/software/rational/offering/testing.html)

*The IBM logo is a registered trademark of IBM in the United States and other countries and used under license. IBM responsibility is limited to IBM products and services and is governed solely by the agreements under which such products and services are provided.*



**McCabe Software** provides Software Quality, Release, and Configuration Management solutions worldwide. McCabe IQ uses a comprehensive collection of software metrics to analyze and visualize the quality and testing of critical applications. McCabe CM utilizes exclusive "Integrated Difference" technology to manage software changes across versions, platforms, and the enterprise. [www.mccabe.com](http://www.mccabe.com)



**MindTree's** testing practice helps you deliver business and technology solutions with world-class quality. When your organization partners with us, it leverages our in-depth experience in test methodologies, frameworks, tools, and processes. MindTree's industry-focused testing services can ensure that your IT applications meet growing business needs with high performance and availability. [www.mindtree.com](http://www.mindtree.com)



**Optimize** is a technology company specializing in software quality assurance and testing solutions. Since its foundation in 2000, Optimize has been serving more than 100 large and mid-sized companies in every major vertical market, providing both software quality assurance and third-party validation. [www.optimize.com.br/english](http://www.optimize.com.br/english)



**Sonata's** testing practice group, with its outcome- /SLA- driven service model, offers a compelling business proposition—cost optimization, productivity enhancement and improved software quality—to enterprises and software product organizations. Sonata has established alliances with testing tool majors and attained expertise in most of the standard test tools. Sonata's testing frameworks and reusable assets help improve the quality of testing and time to market. [www.sonata-software.com/testing\\_services.asp](http://www.sonata-software.com/testing_services.asp)



**Worksoft** is a premier provider of testing solutions to validate end-to-end business processes within the SAP ecosystem. Worksoft speeds SAP software deployments by up to 60%, shortens time to value, and meets compliance, training, and reporting requirements through the auto-generation of documentation and the capture of reusable business process knowledge. [www.worksoft.com](http://www.worksoft.com)

## MEDIA SPONSORS



**PLUS**

**SEE THESE EXHIBITORS AND SPONSORS AT THE EXPO (OCTOBER 1-2)**

**ACM Queue**  
**Agile Project Leadership Network (APLN)**  
 Arsin Corporation, Subsidiary of SemanticSpace Technologies, Ltd.  
 ASTQB  
 AutomatedQA  
**Better Software Magazine**  
**CM Crossroads**  
**CoDe Magazine**  
 Codenomicon, Ltd.

**Cognizant**  
 Collis  
 Critical Logic  
 dynaTrace software  
**HP**  
**IBM**  
**The IEEE Computer Society**  
 ISEB Exams  
 iTKO LISA  
 Klocwork  
**McCabe Software**

**Methods & Tools**  
**MindTree Ltd.**  
 OPNET Technologies  
**Optimize**  
 PushToTest  
 Pragmatic Software  
 QualiSystems  
 Ranorex  
 Reflective Solutions  
 Security Innovation  
 Skytap, Inc.

Software Quality Engineering  
**Sonata Software**  
 SQE Training  
**StickyMinds.com**  
 TCT Computing Group, Inc.  
 TechExcel  
**Twist from Thoughtworks**  
 UST Global, Inc.  
 VenSoft, Inc.  
**WorkSoft**

STARWEST 2008 sponsors are listed in bold. **For Sponsor/Exhibitor news and updates, visit [www.sqe.com/STARWEST](http://www.sqe.com/STARWEST)**



MONDAY, SEPTEMBER 29, 8:30-4:30 (FULL DAY)

## MA Becoming an Influential Test Team Leader

*Randall Rice, Rice Consulting Services, Inc.*

Have you been thrust into the role of test team leader or are you in this role now and want to hone your leadership skills? Test team leadership has many unique challenges, and many test team leaders—especially new ones—find themselves ill-equipped to deal with the problems they face. The test team leader must motivate and support her people while keeping the testing on track within time and budget constraints. Randy Rice focuses on how you can grow as a leader, influence your team and those around you, and impact those outside your team. Learn how to become a person of influence, deal with interpersonal issues, and help your team build their skills and value to the team and the organization. Discover how to communicate your team's value to management, how to stand firm when asked to compromise principles, and how to learn from your successes and failures. Develop your own action plan to become an influential test team leader.



**Randall Rice** is a leading author, speaker, and consultant in the field of software testing and software quality. He has worked with major organizations worldwide to improve the quality of their information systems and to optimize their testing processes. Randall has more than thirty years of experience building and testing mission-critical projects in a variety of environments and has authored many training courses in software testing and software engineering. He is publisher of The Software Quality Advisor newsletter and co-author (with William E. Perry) of *Surviving the Top Ten Challenges of Software Testing*. Randall serves on the board of directors of the American Software Testing Qualifications Board (ASTQB).

## MB Introduction to Systematic Testing

*Dale Perry, Software Quality Engineering*

All too often testers are thrown into the quality assurance/testing process without the knowledge and skills essential to perform the required tasks. To be truly effective, you first must understand what testing is supposed to accomplish and then understand how it relates to the bigger project management and application development picture. After that, you can ask the right questions: What should be tested? How can I design effective and efficient test cases? How much testing is enough? How do I know when I'm finished? How much documentation do I need? Dale Perry explores a testing lifecycle that parallels software development and focuses on defect prevention and early error detection. As Dale shares the basics for implementing a systematic, integrated approach to testing software, learn when, what, and how to test—plus ways to improve the testability of your system.



With more than thirty years experience in information technology, **Dale Perry** has been a programmer/analyst, database administrator, project manager, development manager, tester, and test manager. Dale's project experience includes large systems development and conversions, distributed systems, online applications, both client/server and Web-based. He has been a professional instructor for more than fifteen years and has presented at numerous industry conferences on development and testing. With *Software Quality Engineering* for eleven years, Dale has specialized in training and consulting on testing, inspections and reviews, and other testing and quality related topics.

## MC The Art and Science of SOA Testing

*Rizwan Mallal and Mamoon Yunus, Crosscheck Networks*

Based on today's Web services standards, SOA (Service Oriented Architecture) has ushered in a new era of how applications are designed, developed, tested, and deployed. The promise of SOA to increase development productivity and application flexibility poses new challenges for testers—multiple Web services standards and implementations, legacy applications (of questionable quality) now exposed as Web services, weak or non-existent security controls, and services of possibly diverse origins chained together to create applications. Join Mamoon Yunus and Rizwan Mallal as they lead you through an intensive tutorial that includes hands-on lab work. Roll up your sleeves and dive into the process of testing SOA Web services. Beginning with the Four Pillars of SOA testing, you will learn new concepts to master SOA testing challenges through techniques such as WSDL chaining, schema mutation, and automated filtration. Learn how traditional techniques such as black-, gray-, and white-box testing are applied to SOA testing to maximize test coverage, minimize effort, and release better products.

**Laptop Required.** Admin privileges are also required as software will be installed on them.



**Rizwan Mallal** is the Director of Technology at Crosscheck Networks. A founding member and chief security architect of Forum Systems, Rizwan is responsible for all security related aspects of Forum's technology. Previously, Rizwan was the chief architect at Phobos where he was responsible for developing the industry's first embedded SSL offloader. Before joining Phobos, he was a member of the core engineering group at Raptor Systems, which pioneered the Firewall/VPN space in the mid 1990's.



**Mamoon Yunus** is an advisor to Crosscheck Networks and an industry honored CTO and visionary in Web services-based technologies. As the founder of Forum Systems, Mamoon pioneered Web Services Security Gateways & Firewalls. He has spearheaded Forum's direction and strategy for six generations of award-winning Web Services Security products. Prior to Forum Systems, Mamoon was a global systems engineer for webMethods where he developed XML-based business integration and architecture plans for Global 2000 companies.

## MD Exploratory Software Testing Explained

*Jonathan Kohl, Kohl Concepts, Inc.*

Exploratory testing is an approach to testing that emphasizes the freedom and responsibility of the tester to continually optimize the value of his work. It is the process of three mutually supportive activities performed in parallel: learning, test design, and test execution. With skill and practice, exploratory testers typically uncover an order of magnitude more problems than the same amount of effort spent on procedurally scripted testing. All testers conduct exploratory testing in one way or another, but few know how to do it systematically to obtain the greatest benefits. Even fewer testers can articulate the process. Jonathan Kohl describes specific heuristics and techniques of exploratory testing to help you get the most from this highly productive approach. Jonathan focuses on the skills and dynamics of exploratory testing itself and how it can be combined with scripted approaches. (For insight into how to manage and measure ET, attend Jonathan Bach's tutorial on Session-Based Exploratory Testing.)

**Laptop Required.** This is a hands-on course. A laptop—preferably with Microsoft Windows capability—is required for some of the exercises.



**Jonathan Kohl** is the founder and principal software testing consultant with Kohl Concepts, Inc., based in Calgary, Alberta, Canada. A noted testing thinker, Jonathan is recognized as a leader in the exploratory testing community. He is a popular author and speaker who believes that testing is a challenging intellectual craft. Jonathan's blog on software development and testing issues is one of the most often read testing blogs in the industry. A regular contributor to *Better Software* magazine, Jonathan was a guest Technical Editor for the March 2007 issue.

**ME** **How to Teach Yourself Testing** **NEW***James Bach, Satisfice, Inc.*

Are you in control of your testing education? Do you have a systematic approach to learning the skills a great tester needs? James Bach shares his personal system of testing self-education. It's a system based on analyzing personal experiences and questioning conventional wisdom. He explains and demonstrates the methods that he has used to develop context-driven testing ideas since 1987. You can use similar methods to draw out and codify the lessons of your own experiences. James discusses how to sort through the differing schools of testing; the entry points for personal testing education; a syllabus of software testing concepts; how to identify, articulate, and test your own heuristics; and how to assess your progress. Whether you are new to testing, working to be a great test lead, or want to become a better testing consultant, this tutorial will take you down the road to more effective learning.



**James Bach** is founder and principal consultant of Satisfice, Inc., a software testing and quality assurance company. In the '80s, James cut his teeth as a programmer, tester, and SQA manager in Silicon Valley in the world of market-driven software development. For nearly ten years, he has traveled the world teaching rapid software testing skills and serving as an expert witness on court cases involving software testing. James is the author of *Lessons Learned in Software Testing and Secrets of a Buccaneer Scholar*, which will be published in spring 2009.

**MF** **Transition to Agile Development: A Tester's View***Bob Hartman, Net Objectives*

Adopting an agile development methodology changes many familiar practices for both developers and testers. Join Bob Hartman to examine the challenges many testers face as agile development practices move into the mainstream and into their organizations. Teams new to agile or exploring agile practices have discovered that the transition from traditional testing practices to the lean-agile "test first" approach is a significant challenge for the development team and, in particular, for test engineers. Learn how requirements practices and documents differ when the team is using agile development practices. Find out about new workflows needed for test development and execution, and process changes for tracking and repairing defects. Discover how faster release schedules can affect testing and the entire team. Using case studies—both successes and failures—Bob discusses transition strategies and solutions for test and development teams. Learn from these experiences and apply their lessons to the challenges you may face as you enter the land of agile development.



With more than thirty years of experience developing software, **Bob Hartman** has served in almost every role in the software industry including developer, tester, documentation writer, trainer, project manager, business analyst, development manager, and executive. He brings a logic-based approach to software development and quality. Bob is known for having a unique talent for breaking software within the first ten minutes of using it and brings that unique insight into the testing area. He is a Certified ScrumMaster (CSM) and Certified Scrum Practitioner (CSP) who does training and coaching in agile development. Bob teaches courses on Lean Agile Testing, Implementing Scrum, and Lean Software Development among others.

**MG** **Scripting Techniques for Testers** **NEW***Dion Johnson, DiJohn Innovative Consulting, Inc.*

Automating functional tests for highly dynamic applications is a daunting task. Unfortunately, most testers rely on automation tools that produce static test suites that are difficult and expensive to change. With complex automation frameworks and expensive testing tools, it is no wonder that automated testing often fails to live up to its promise. But, there is another way that is simple and almost free! By learning basic scripting language skills, you can begin immediately to automate time-consuming, everyday testing tasks. Scripting saves valuable time doing repetitive tasks so that you can focus on more important work. Using the Ruby scripting language and Internet Explorer, you will practice scripted automation techniques on an HTML application. These techniques address many of your test automation needs, including dynamic data creation, automated input entry, and exception handling—all of which can increase the coverage, maintainability, scalability, and robustness of your tests. *Participants should have scripting experience or knowledge of basic programming control-flow statements and logic—if-then-else, for-next, etc.*

**Laptop Required.** Be sure to bring your Windows laptop with Internet Explorer and Excel. Because working in pairs is encouraged, feel free to bring a friend to share your PC.



**Dion Johnson** has more than thirteen years of experience providing IT services to both government and private industry. He has spent much of his professional career as a consultant, tasked with handling all aspects of the delivery of on-site customer services, particularly in the areas of quality assurance, quality control, software process improvement, and requirements analysis. Dion has delivered award-winning and highly acclaimed presentations at many of the most prestigious industry conferences, including STAREAST, STARWEST, and the Better Software Conference & EXPO. Dion also writes for Better Software magazine and StickyMinds.com.

**MH** **Key Test Design Techniques***Lee Copeland, Software Quality Engineering*

All testers know that we can create many more test cases than we will ever have time to create and execute. The major problem in testing is choosing a small, "smart" subset from the almost infinite number of possibilities available. Join Lee Copeland to discover how to design test cases using formal black-box techniques, including equivalence class and boundary value testing, decision tables, state-transition diagrams, and all-pairs testing. Also, explore white-box techniques and their associated coverage metrics. Evaluate more informal approaches, such as random and hunch-based testing, and learn about the importance of exploratory testing to enhance your testing ability. Choose the right test case documentation format for your organization. Use the test execution results to continually improve your test designs.



**Lee Copeland** has more than thirty-five years of experience as a consultant, instructor, author, and information systems professional. He has held a number of technical and managerial positions with commercial and non-profit organizations in the areas of applications development, software testing, and software development process improvement. Lee frequently speaks at software conferences both in the US and internationally and currently serves as Program Chair for the Better Software Conference & EXPO, the STAR testing conferences, and Software Quality Engineering's new Agile Development Practices conference. Lee is the author of *A Practitioner's Guide to Software Test Design*, a compendium of the most effective methods of test case design.

**Over 98% of 2007 Attendees  
Recommend STARWEST  
to Others in the Industry**

### MI **Becoming a Trusted Advisor to Senior Management** NEW

Lloyd Roden, Grove Consultants

How can test managers present information about test results so that the decision-makers receive the correct message? Testing generates a huge amount of raw data, which must be analyzed, processed, summarized, and presented to management so the best decisions can be made quickly. Using his experiences as a test manager and consultant, Lloyd Roden shares ways to communicate with and disseminate information to management. Develop your skills so you become a "trusted advisor" to senior management rather than the "bearer of bad news." Discover innovative ways to keep the information flowing to and from management and avoid losing control of the test process, particularly near the delivery date. Learn how to deal effectively with various controversies that prevent senior managers from taking us seriously.



With more than twenty-five years in the software industry, **Lloyd Roden** has worked as a developer, managed an independent test group within a software house, and joined Grove Consultants in 1999. Lloyd has been a speaker at STAREAST, STARWEST, EuroSTAR, AsiaSTAR, Software Test Automation, Test Congress, and Unicom conferences as well as Special Interest Groups in software testing in several countries. He was program chair for both the tenth and eleventh EuroSTAR conferences.

### MJ **Reliable Test Effort Estimation**

Ruud Teunissen, POLTEQ IT Services BV

How do you estimate your test effort? And how reliable is that estimate? Ruud Teunissen presents a practical and useful test estimation technique directly related to the maturity of your test and development process. A reliable effort estimation approach requires five basic elements: (1) Strategy – Determine what to test (performance, functionality, etc.) and how thoroughly it must be tested. (2) Size – Yes, it does matter—not only the size of the system but also the scope of your tests. (3) Expected Quality – What factors have been established to define quality? (4) Infrastructure and Tools – Define how fast you can test. Without the proper organizational support and the necessary tools, you'll need more time. (5) Productivity – How experienced and efficient is your team? While it's fun to learn new techniques, it means your time is not being spent finding defects.



**Ruud Teunissen**, International Test Consultant at POLTEQ IT Services BV, has performed several test functions in a large number of IT projects: tester, test specialist, test consultant, and test manager. Together with Martin Pol, he is co-author of several books on structured testing. His main focus at this moment is test management and test process improvement.

### MK **Using Visual Models for Test Case Design**

Rob Sabourin, AmiBug.com, Inc.

Designing test cases is a fundamental skill that all testers should master. Rob Sabourin shares a graphical technique he has employed to design powerful test cases that will surface important bugs quickly. These skills can be used in exploratory, agile, or engineered contexts—anytime you are having problems designing a test. Rob illustrates how you can use Mindmaps to visualize test designs and better understand variables being tested, one-at-a-time and in complex combinations with other variables. He presents the Application-Input-Memory (AIM) heuristic through a series of interactive exercises. We'll use a widely available free, open-source tool called FreeMind to help implement great test cases and focus our testing on what matters to quickly isolate critical bugs. If you are new to testing, these techniques will remove some of the mystery of good test case design. If you're a veteran tester, these techniques will sharpen your skills and give you some new test design approaches.

*Participants are encouraged to bring a laptop computer to this session.*



**Rob Sabourin** has more than twenty-five years of management experience, leading teams of software development professionals. A well-respected member of the software engineering community, Rob has managed, trained, mentored, and coached hundreds of top professionals in the field. He frequently speaks at conferences and writes on software engineering, SQA, testing, management, and internationalization. The author of *I am a Bug!*, the popular software testing children's book, Rob is an adjunct professor of Software Engineering at McGill University.

### ML **Measurement and Metrics for Test Managers**

Rick Craig, Software Quality Engineering

To be most effective, test managers must develop and use metrics to help direct the testing effort and make informed recommendations about the software's release readiness and associated risks. Because one important testing activity is to "measure" the quality of the software, test managers must measure the results of both the development and testing processes. Collecting, analyzing, and using metrics is complicated because many developers and testers feel that the metrics will be used "against them." Rick Craig addresses common metrics: measures of product quality, defect removal efficiency, defect density, defect arrival rate, and testing status. Rick offers guidelines for developing a test measurement program, rules of thumb for collecting data, and ways to avoid "metrics dysfunction." Various metrics paradigms, including Goal-Question-Metric, are addressed with a discussion of the pros and cons of each. *Participants are urged to bring their metrics problems and issues for use as discussion points.*



**Rick Craig** is a consultant, lecturer, author, and test manager, who has led numerous teams of testers on both large and small projects. In his twenty-five years of consulting worldwide, Rick has advised and supported a diverse group of organizations on many testing and test management issues. From large insurance providers and telecommunications companies to smaller software services companies, he has mentored senior software managers and helped test teams improve their effectiveness. Rick is co-author of *Systematic Software Testing* and a frequent speaker at testing conferences, including every STAR conference since its inception.

### MM **Difficult Testing Questions and How to Answer Them** NEW

Michael Bolton, DevelopSense

Testers live in a world of great complexity, scarce information, and extraordinary time pressures. With all these challenges, really good testing is less about confirming, verifying, and validating, and more about thinking, questioning, exploring, investigating, and discovering. While technical testing skills are important, you need better thinking skills to solve your biggest testing questions. Michael Bolton teaches you the skills—questioning, critical thinking, context-driven analysis, and general systems thinking—that can help you deal confidently and thoughtfully with your testing challenges. In this interactive workshop, you will examine common cognitive biases within testing and practice the thinking tools you need to overcome them. You'll learn to use modeling and general systems approaches to manage complexity and see more clearly. Work with Michael and others to explore your most difficult testing questions—and find innovative approaches to answer them.

*Participants are encouraged to bring a laptop computer to this session.*



**Michael Bolton** has been teaching software testing on five continents for eight years. He is the co-author (with senior author James Bach) of *Rapid Software Testing*, a course that presents a methodology and mindset for testing software expertly in uncertain conditions and under extreme time pressure. Michael is a co-founder of the *Toronto Workshops on Software Testing*. He has a regular column in *Better Software* magazine, writes for *Quality Software*, and sporadically produces his own newsletter. Michael lives in Toronto, Canada, with his wife and two children. He can be reached at [mb@developsense.com](mailto:mb@developsense.com), or through his Web site, <http://www.developsense.com>.



**MO Risk-Based Testing***Julie Gardiner, Grove Consultants*

Risks are endemic in every phase of every project. One key to project success is to identify, understand, and manage these risks effectively. However, risk management is not the sole domain of the project manager, particularly with regard to product quality. It is here that the effective tester can significantly influence the project outcome. Julie Gardiner explains how risk-based testing can shape the quality of the delivered product in spite of such time constraints. Join Julie as she reveals how you can apply product risk management to a variety of organizational, technology, project, and skills challenges. Through interactive exercises, receive practical advice on how to apply risk management techniques throughout the testing lifecycle—from planning through execution and reporting. Take back a practical process and the tools you need to apply risk analysis to testing in your organization.



With more than eighteen years of experience in the IT industry, **Julie Gardiner** has spent time as an analyst programmer, Oracle DBA, and project manager. She has first-hand experience as a test analyst, test team leader, test consultant, and test manager. At Grove Consultants, Julie provides consultancy and training in all aspects of testing, specializing in risk-based testing, agile testing, test management, and people issues. She is a certified ScrumMaster. Julie won best presentation at STAREAST 2007 and 2005; best presentation at BCS SIGIST 2005; and best tutorial at EuroSTAR 2006.

**MO Spend Wisely, Test Well: Making a Financial Case for Testing***Susan Herrick, EDS-Global Testing Practice*

Organizations that develop software always profess absolute commitment to product quality and customer satisfaction. At the same time, they often believe that “all that testing isn’t really necessary.” Test managers must be able to quantify the financial value of testing and substantiate their claims with empirical data. Susan Herrick provides experienced test managers with quantitative approaches to dispel the prevailing myths about the negative bottom-line impact of testing, make a compelling business case for testing throughout the project lifecycle, and provide decision-makers with information that allows them to make fiscally responsible choices about test efforts. During a hands-on activity, you will calculate, analyze, and substantiate answers to such questions as, “What will it cost if we don’t test at all?” “Should we rely on the system and acceptance testers to find all the defects?” “Can our experienced developers test their own code?” and “Should experienced users perform the acceptance testing?” Answer these and more questions with the numbers at hand to back up your claims.

*To benefit fully from the hands-on activity, each participant should bring a laptop. All participants will receive as a takeaway a CD containing a calculation tool (with full instructions).*



With twenty-five years of involvement in the development of IT solutions and fifteen years of testing experience, **Susan Herrick** is currently a testing consultant and corporate leader of Testing Management and Consulting for the Global Testing Capability at EDS. She provides expertise, leadership, and guidance in “architecting testing solutions,” particularly in the areas of testing strategy development and testing management and measurement. Susan has contributed to the development of supporting processes, tools, and techniques in these key areas.

**MP To Infinity and Beyond: Extreme Boundary Testing***Rob Sabourin, AmiBug.com*

If you think you have already explored all of the important boundaries as part of your testing, this dynamic, interactive presentation will open your eyes to some often-missed edges and offer you great techniques to expose and explore them. You’ll dive into the rich universe of boundaries related to systems behavior, environments, system limits, design limitations, and even eccentric user behaviors. Rob Sabourin helps you learn to see and explore the final frontiers of your software and look beyond the confines of common knowledge to see the aliens and strange monsters lurking. In this hands-on workshop, you’ll participate in a series of fun, interactive exercises and experience rich boundary examples from Rob’s recent projects. Practice identifying and exercising the data conditions that influence a system’s behavior and understand how critical values lead to emergent behaviors, which can make or break software projects. In addition to practicing traditional boundaries value analysis and equivalence partitioning techniques, you will learn about exploratory testing, failure mode analysis, and several stress testing experiments you can perform.



**Rob Sabourin** has more than twenty-five years of management experience, leading teams of software development professionals. A well-respected member of the software engineering community, Rob has managed, trained, mentored, and coached hundreds of top professionals in the field. He frequently speaks at conferences and writes on software engineering, SQA, testing, management, and internationalization. The author of *I am a Bug!*, the popular software testing children’s book, Rob is an adjunct professor of Software Engineering at McGill University.

**MQ The Craft of Bug Investigation***Jon Bach, Quardev, Inc.*

At testing conferences, many presentations mention techniques and processes meant to help you find bugs, but few talk about what to do when you actually find one. If it’s as simple as filing a report about what you saw, how do you know that’s the real problem? What do you do when you file a bug, but the developer wants you to give him more information? How do you reproduce pesky, intermittent bugs that come in from customers? Join Jon Bach in this hands-on tutorial to help you practice investigation and analysis skills such as questioning, conjecturing, branching, and backtracking that might help you unearth more context about the problems. If you’re telling stories about the bug that got away, this tutorial gives you the opportunity to try some techniques that may trap it so you can earn more credibility, respect, and autonomy from your stakeholders. *Collaboration is encouraged during the session, so bring your tool suggestions, tester’s notebook, and scientific mindset.*



**Jon Bach** is senior consultant and manager for corporate intellect at Quardev, Inc., a Seattle outsource test lab where he manages testing projects ranging from a few days to several months using Rapid Testing techniques. In 2000, Jon and his brother James invented the “Session-Based Test Management” technique for managing and measuring exploratory testing. In his thirteen years of testing, Jon has been a test contractor, full-time test manager, and consultant for companies such as Microsoft and Hewlett-Packard. He has written articles for both *Better Software* and *IEEE Computer* magazines.

**MR Managing Test Outsourcing***Martin Pol, POLTEQ IT Services BV***UPDATED**

When outsourcing all or part of your testing efforts to a third-party vendor, you need a special approach to make testing effective and controlled. Martin Pol explains his roadmap to successful outsourcing and offers ways to define the objectives, the strategy, and the scope (what tasks should be outsourced and what tasks should not—at least not yet). He describes how to select your supplier and how to migrate, implement, and cope with organizational issues. Martin discusses contracts, service levels, and ways to monitor and control tasks. He focuses on a technique for scoping the project, defining service levels, and establishing a specific set of metrics. The good news for testers is that outsourcing requires more testing—not less—and that new testing jobs are coming into existence. Testing the outsourcing is becoming a very important control mechanism for outsourcing in general.



**Martin Pol** has played a significant role in helping to raise the awareness and improve the performance of testing worldwide. Martin provides international testing consulting services through POLTEQ IT Services BV. During recent years, he has specialized in test outsourcing/offshoring, and he has developed an approach to successfully deal with this phenomenon. His experiences in both India and China are of great value. He has supported many organizations to define the test service levels, to organize the prerequisites, and to implement test outsourcing management and monitoring.

### TA Adapting to Agile

*Elisabeth Hendrickson, Quality Tree Software, Inc.*

When a development team adopts an agile process such as Scrum or XP, testers find that their traditional practices no longer fit. The extensive up-front test planning and heavyweight test documentation used in traditional development environments just get in the way in an agile world. In this experiential workshop, you experience the transition to agile through a paper-based simulation (no programming required). In a series of iterations, the team attempts to deliver a product that the customer is willing to buy, thus generating revenue for the company. As with real projects, producing a working product on a tight schedule can be challenging. After each iteration, your team reflects on key events and adjusts to increase productivity for the next iteration. Learn to apply the principles of visibility, feedback, communication, and collaboration to increase the team's rate of delivery. By the end of the workshop, you will have an intuitive understanding of agile and, in particular, the shifting role of Test/QA in agile development.



In the software industry since 1984, **Elisabeth Hendrickson** has held positions as a tester, programmer, test automation manager, quality engineering director, and technical writer working for companies ranging from a twenty-person startup to a large multi-national software vendor. In 2003, Elisabeth became involved with the agile community, became a Certified ScrumMaster, and in 2006 joined the board of directors of the Agile Alliance. Today, Elisabeth spends her time teaching, speaking, writing, and working on Extreme Programming teams with test-infected programmers who value her obsession with testing.

### TB How to Build, Support, and Add Value to Your Test Team

*Lloyd Roden and Julie Gardiner, Grove Consultants*

As a new or current test manager, you may have many questions—How do I create a new team? How can I help my current team become more efficient and effective? How can I build my organization's confidence in our work? How can I find needed resources? Based on a people-oriented—rather than task-oriented—approach to software testing, Lloyd Roden and Julie Gardiner describe how to build and retain successful test teams. Discover the characteristics of successful testers and test managers. Identify the qualities you should look for to recruit the right people. Learn what you must do for your team and what they should do for themselves. Discuss how to promote the value of testing within the organization while building good working relationships with developers and other organizations. Discuss these relevant issues with others facing the same challenges. Lloyd and Julie provide utilities, spreadsheets, and templates to help you become a successful test manager.



With more than twenty-five years in the software industry, **Lloyd Roden** has worked as a developer, managed an independent test group within a software house, and joined Grove Consultants in 1999. Lloyd has been a speaker at STAREAST, STARWEST, EuroSTAR, AsiaSTAR, Software Test Automation, Test Congress, and Unicom conferences as well as Special Interest Groups in software testing in several countries. He was program chair for both the tenth and eleventh EuroSTAR conferences.



With more than eighteen years of experience in the IT industry, **Julie Gardiner** has spent time as an analyst programmer, Oracle DBA, and project manager. She has first-hand experience as a test analyst, test team leader, test consultant, and test manager. At Grove Consultants, Julie provides consultancy and training in all aspects of testing, specializing in risk-based testing, agile testing, test management, and people issues. She is a certified ScrumMaster. Julie won best presentation at STAREAST 2007 and 2005; best presentation at BCS SIGIST 2005; and best tutorial at EuroSTAR 2006.

### TC Essential Test Management and Planning

*Rick Craig, Software Quality Engineering*

The key to successful testing is effective and timely planning. Rick Craig introduces proven test planning methods and techniques, including the Master Test Plan and level-specific test plans for acceptance, system, integration, and unit testing. Rick explains how to customize an IEEE-829-style test plan and test summary report to fit your organization's needs. Learn how to manage test activities, estimate test efforts, and achieve buy-in. Discover a practical risk analysis technique to prioritize your testing and help you become more effective with limited resources. Rick offers test measurement and reporting recommendations for monitoring the testing process. Discover new methods and develop renewed energy for taking test management to the next level in your organization.



**Rick Craig** is a consultant, lecturer, author, and test manager, who has led numerous teams of testers on both large and small projects. In his twenty-five years of consulting worldwide, Rick has advised and supported a diverse group of organizations on many testing and test management issues. From large insurance providers and telecommunications companies to smaller software services companies, he has mentored senior software managers and helped test teams improve their effectiveness. Rick is co-author of Systematic Software Testing and a frequent speaker at testing conferences, including every STAR conference since its inception.

### TD Understanding Software Performance Testing

*Dale Perry, Software Quality Engineering*

What does it take to properly plan and implement a performance test? What factors need to be considered? What is your performance test tool telling you? Do you really need a performance test? Is it worth the cost? These questions plague all performance testers. In addition, many performance tests do not appear to be worth the time it takes to run them, and the results never seem to resemble—yet alone predict—production system behavior. Performance tests are some of the most difficult tests to create and run, and most organizations don't fully appreciate the time and effort required to properly execute them. Dale Perry discusses the key issues and realities of performance testing—what can and cannot be done with a performance test, what is required to do a performance test, and what the test "really" tells you.



With more than thirty years experience in information technology, **Dale Perry** has been a programmer/analyst, database administrator, project manager, development manager, tester, and test manager. Dale's project experience includes large systems development and conversions, distributed systems, online applications, both client/server and Web-based. He has been a professional instructor for more than fifteen years and has presented at numerous industry conferences on development and testing. With Software Quality Engineering for eleven years, Dale has specialized in training and consulting on testing, inspections and reviews, and other testing and quality related topics.

### TE Just-In-Time Testing

*Rob Sabourin, AmiBug.com, Inc.*

Turbulent Web development and other market-driven projects experience almost daily requirements modifications, changes to user interfaces, and the continual integration of new functions, features, and technologies. Rob Sabourin shares proven, practical techniques to keep your testing efforts on track while reacting to fast-paced projects with changing priorities, technologies, and user needs. Rob covers test planning techniques and organization strategies, scheduling and tracking, blending scripted and exploratory testing, identifying key project workflows, and using testing and test management tools. Learn how to create key decision-making workflows for test prioritization and bug triage, adapt testing focus as priorities change, identify technical risks, and respect business importance. Come away with a new perspective on your testing challenges and discover ways to take control of the situation—rather than to be controlled by it.



**Rob Sabourin** has more than twenty-five years of management experience, leading teams of software development professionals. A well-respected member of the software engineering community, Rob has managed, trained, mentored, and coached hundreds of top professionals in the field. He frequently speaks at conferences and writes on software engineering, SQA, testing, management, and internationalization. The author of I am a Bug!, the popular software testing children's book, Rob is an adjunct professor of Software Engineering at McGill University.

**TF Requirements-Based Testing***Richard Bender, Bender RBT, Inc.*

Testers use requirements as an oracle to verify the success or failure of their tests. Richard Bender presents the principles of the Requirements-Based Testing methodology in which the software's specifications drive the testing process. Richard discusses proven techniques that ensure requirements are accurate, complete, unambiguous, and logically consistent. Requirements-based testing provides a process for first testing the integrity of the specifications. It then provides the algorithms for designing an optimized set of tests sufficient to verify the system from a black-box perspective. Find out how to design test cases to validate that the design and code fully implement all functional requirements. Determine which test design strategy—cause-effect graphing, equivalence class testing, orthogonal pairs, and more—to apply to your applications. By employing a requirements-based testing approach, you will be able to quantify test completion criteria and measure test status.



**Richard Bender** has been involved in test and evaluation since 1969. He has authored and co-authored books and courses on quality assurance and test, software development lifecycles, analysis and design, software maintenance, and project management. Richard has worked with an international clientele in a wide range of industries from financial to academic.

**TG Session-Based Exploratory Testing***Jon Bach, Quardev, Inc.*

The agile nature of exploratory testing makes it a widely-used and effective test approach, especially when testing time is limited. But despite the ability of testers to rapidly apply their skill, exploratory testing is often dismissed by project managers who regard exploration as unreproducible, immeasurable, and unaccountable. If you find this to be true where you work, a solution may be to use Session-Based Test Management (SBTM), developed by Jon Bach and his brother James, to solve these problems. In SBTM, testers are assigned areas of a product to explore, and testing is time-boxed in "sessions" which have mission statements called "charters." Together, these create a meaningful and countable unit of work. Using a simulated project, you'll practice elements of sessions, including chartering, paired testing (working with another tester on the same mission), storytelling (taking notes during your testing), and debriefing (responding to questions after your session). Jon will use a freely available, open source tool to help manage and measure testing effort done in sessions.



**Jon Bach** is senior consultant and manager for corporate intellect at Quardev, Inc., a Seattle outsource test lab where he manages testing projects ranging from a few days to several months using Rapid Testing techniques. In 2000, Jon and his brother James invented the "Session-Based Test Management" technique for managing and measuring exploratory testing. In his thirteen years of testing, Jon has been a test contractor, full-time test manager, and consultant for companies such as Microsoft and Hewlett-Packard. He has written articles for both Better Software and IEEE Computer magazines.

**TH Test Process Improvement** **UPDATED***Martin Pol and Ruud Teunissen, POLTEQ IT Services BV*

What is the maturity of your testing process? How do you compare to other organizations and to industry standards? To find out, join Martin Pol and Ruud Teunissen for an introduction to the Test Process Improvement (TPI®) model, an industry standard for testing maturity assessments. Although many organizations want to improve testing, they lack the foundations required for success. Improving your testing requires three things: (1) understanding key test process areas, (2) knowing your current position in each of these areas, and (3) having the tools and skills to implement needed improvements. Rather than guessing what to do, begin with the TPI® model as your guide. Using as examples real world TPI® assessments that they have performed, Martin and Ruud describe a practical assessment approach that is suitable for both smaller, informal organizations and larger, formal companies. Take back valuable references, templates, examples, and links to start your improvement program.

TPI® is a registered trademark of Sogeti USA LLC.



**Martin Pol** has played a significant role in helping to raise the awareness and improve the performance of testing worldwide. Martin provides international testing consulting services through POLTEQ IT Services BV. During recent years, he has specialized in test outsourcing/offshoring, and he has developed an approach to successfully deal with this phenomenon. His experiences in both India and China are of great value. He has supported many organizations to define the test service levels, to organize the prerequisites, and to implement test outsourcing management and monitoring.



**Ruud Teunissen**, International Test Consultant at POLTEQ IT Services BV, has performed several test functions in a large number of IT projects: tester, test specialist, test consultant, and test manager. Together with Martin Pol, he is co-author of several books on structured testing. His main focus at this moment is test management and test process improvement.

**II Protecting Your Applications from Web Security Vulnerabilities** **NEW***Caleb Sima, Hewlett Packard*

Does your security testing focus mainly on user identification, access control, and encryption? Although that's a start, you also should be concerned about application security from the outside world of dangerous hackers. Caleb Sima, a white-hat hacker who has broken into countless Web applications, demonstrates popular hacking techniques, such as SQL injection, cross-site scripting, and more. Using live Web sites, he takes you step-by-step through traditional Web site and newer Ajax security vulnerabilities. Learn where these issues are present in your systems, how you can find them, and what hackers can accomplish if you don't. Caleb describes attacks via server-side application "holes" and how phishers, boters, and worm authors use these vulnerabilities to exploit Web-based systems. He discusses browser/server interaction issues, the increasing attack surface in newer Web applications, repudiation of HTTP requests, and how hackers expose application logic. Get a "behind the scenes" look at the thought processes of hackers who are actively working to circumvent your Web applications' security measures.



**Caleb Sima** is HP Application Security Center's chief technologist. He is the former co-founder and CTO of SPI Dynamics acquired by HP Software in August 2007. He is responsible for directing the strategic direction of the company's Web application security solutions. Caleb is widely recognized as an expert in Web security, penetration testing, and identifying emerging security threats. His pioneering efforts and expertise in Web security have helped define the direction of the Web application security industry. Caleb is a contributing author to various magazines and online columns, and is a co-author of the book, *Hacking Exposed Web Applications: Web Security Secrets & Solutions*.



### TJ Test Automation Patterns: Best Practices and Common Pitfalls

Gerard Meszaros, Independent Consultant

The extensive use of automated testing has been a breakthrough in improving the quality of software. By now, many companies have experimented with automating functional tests and, perhaps, unit tests. Those companies that have had good experiences rave about automation and cannot imagine having been successful without it. However, for every success story, there are many untold stories of disappointment. What separates the successes from the disappointments? Join Gerard Meszaros as he describes common problems encountered when writing and running automated tests. He characterizes the problems in the form of their visible symptoms, discusses their root causes, and suggests possible solutions expressed in the form of patterns that have worked for others. Many of these causes and patterns are equally applicable to unit tests using xUnit, to automated functional and acceptance tests using tools such as Watir, and to record and playback test tools such as Mercury's QuickTest. Gerard illustrates these concepts with demonstrations and short, hands-on exercises.



A Calgary, Canada-based consultant and trainer, **Gerard Meszaros** specializes in agile development processes. He has more than twenty-five years of experience building and testing software intensive systems in both product development and IT environments with technologies ranging from Java and .Net to Ruby and SAP's ABAP. Gerard coaches cross-functional teams as they learn how to better envision, specify, develop, and test software systems using agile methods. He is a frequent speaker at major international software conferences and is the author of xUnit Test Patterns—Refactoring Test Code.

### TK High Impact Software Inspections

Ed Weller, Integrated Productivity Solutions, LLC

Software inspections were first formally developed at IBM in 1972. More than three decades later, inspections remain relevant, and more importantly, they are feasible and will work in most environments. Ed Weller has successfully initiated numerous inspection programs that have stood the test of time. His experience provides the practical basis for this tutorial covering the economics of inspections and how they can improve the bottom line; the roles within the inspection process and why they are important to success; the steps in the process and how to measure their effectiveness; measurements needed to evaluate success and point out areas for improvement; the relationship of inspections to unit testing; and the impact of the global workforce on inspections and tools necessary to adapt inspections to multiple locations and time zones. Join Ed for this inspection process overview and learn more about the six critical factors you need to consider when you are thinking about or planning to implement inspections.

CMMI® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University  
SCAMPF™ is a service mark of Carnegie Mellon University



**Ed Weller** has more than forty years of experience in hardware, test, software, systems, and software process engineering. His primary focus for the past fifteen years has been on software process and metrics. Ed is the principal of Integrated Productivity Solutions, LLC, a consulting company focused on improving quality and productivity. Ed is an SEI-Certified SCAMPF™ High Maturity Lead Appraiser for SCAMPF™ V1.2 and instructor for the Introduction to the CMMI®. Ed has given nearly fifty presentations and tutorials at conferences. Ed can be reached at ed.weller@integratedproductivitysolutions.com.

### TL Agile Defect Reporting With User Stories and Tests NEW

Antony Marciano, testingReflections.com

If you are part of a team that employs user stories or acceptance tests as a way to document software requirements, this presentation is for you. Join Antony Marciano to learn how to use these same tools for reporting incidents and bugs you find. Whereas user stories and acceptance tests describe the system's desired behavior, defect reports describe misbehavior. And behind each misbehavior is a desired behavior—a hidden user story. Every agile development team has their unimplemented user stories held in the product backlog. Hiding in the shadows is the secret backlog—your defect list. As the project progresses, the secret backlog of defects can grow, never receiving as much attention as the new user stories. Through a series of exercises to build the participant's user-story and acceptance test writing skills, Antony shows how these can be applied to represent defect reports as acceptance tests. This enables you to reverse engineer the hidden user story—helping to ensure that defects get the attention they deserve.



**Antony Marciano** has more than thirteen years of experience in software testing across numerous sectors, including mobile and fixed telecommunications, banking, publishing, broadcasting, advertising, law, and education. Since 2000, much of Antony's work has been on agile projects. Now, as a practitioner, mentor, coach, and consultant, he helps teams realize the benefits associated with agile development. Antony is creator and curator of testingReflections.com, one of the most influential software testing sites on the Internet and is a Technical Editor for Better Software magazine.

### TM AJAX Testing: Inside and Out NEW

Paco Hope, Cigital

AJAX—Asynchronous, JavaScript, and XML—is a modern application development technique that allows a Web-based application to look and feel just like a full-fledged desktop or client/server program. AJAX applications pose unique testing challenges because so much of the application's logic runs inside the Web browser. To thoroughly test applications employing AJAX techniques, you need to understand the technology and adopt specific testing approaches and special testing tools. Paco Hope presents a short introduction to dynamic HTML, JSON, and the core technologies that make AJAX possible. Then, he explores the approaches required to adequately test AJAX applications—from the outside-in and the inside-out. Paco demonstrates an AJAX application and shows you his strategies to test it. Finally, he discusses trade-offs of different testing tools—some open source and some commercial—that enable you to interactively and automatically test AJAX.



A managing consultant at Cigital, **Paco Hope** has more than twelve years of experience in software and operating system security. His areas of expertise include software security policy, code analysis, host security, and PKI. Paco has worked extensively with embedded systems in the gaming and mobile communications industries and has served as a subject matter expert on issues of network security standards in the financial industry. Paco is co-author of Mastering FreeBSD and OpenBSD Security. Prior to joining Cigital, he served as director of product development for Tovarish, Inc.



TUESDAY, SEPTEMBER 30, 1:00-4:30 (HALF DAY - PM)

**TN** **Introducing Keyword-Driven Test Automation** **NEW***Hans Buwalda, LogiGear Corporation*

Keyword-driven test automation has entered the mainstream of testing. Keywords is a powerful approach for you to reach a high level of automation with the lowest possible effort. This approach can bring you the flexibility, manageability, and maintainability that test automation demands. Hans Buwalda introduces keyword-driven test automation, based on his successful Action Based Testing methodology. Then, he discusses how you can implement keywords by having good test design techniques, a workable automation architecture, and the management skills for success. When properly implemented, keyword-driven test automation projects can result in high automation percentages with significant reusability and low maintenance. Learn how you can start on the path to meeting Hans' "5% challenges" for test automation: No more than 5% of your test cases should be executed manually, and no more than 5% of your total testing effort should be used to achieve this automation.



**Hans Buwalda** is an internationally recognized expert in test development and testing technology management and a pioneer of keyword-driven test automation. He was the first to present this approach, which is now widely used throughout the testing industry. Originally from the Netherlands, Hans now lives and works in California as CTO of LogiGear Corporation, directing the development of what has become the successful Action Based Testing™ methodology for test automation and its supporting TestArchitect™ toolset. Prior to joining LogiGear, Hans served as project director at CMG (now Logica) in the Netherlands. He is co-author of Integrated Test Design and Automation and a frequent speaker at international conferences.

**TO** **Essential Thinking Skills for Testers and Test Managers***Krishna Iyer and Mukesh Mulchandani, ZenTEST Labs*

The most important skills that testers need in their work are thinking skills. While often ignored in favor of testing techniques and automation tools, improving testers' thinking skills has the greatest benefit. Having trained more than 5,000 testers in testing skills and more than 500 testers in essential thinking skills, Krishna Iyer and Mukesh Mulchandani have proven this fact. They present three vital thinking skills—critical thinking, creative thinking, and coverage thinking. Designed for both testers and test managers, this class helps you develop an eye to see what no one else sees, a nose to sniff out more defects, and an ear to critically evaluate every claim you hear. Join Krishna and Mukesh for the latest research in cognitive thinking; learn practical techniques such as ideational fluency, test mapping, and filtering bias; and understand the mindset of effective testers.



**Krishna Iyer**, CEO of ZenTEST Labs, is a young entrepreneur, a prolific speaker, and author. Before ZenTEST Labs, Krishna was a quality manager at Kanbay where he worked with clients such as CitiFinancial, HSBC, IBM, and GE. Krishna shapes ZenTEST Labs strategy using his financial background, improves ZenTEST Labs operations using his rich IT and process consulting experience, and transforms its culture using his expertise as a behavioral trainer. Krishna is a regular presenter at testing and quality conferences including STARWEST and STAREAST.



**Mukesh Mulchandani**, CTO of ZenTEST Labs, is responsible for establishing ZenTEST Labs as a key player in the software testing domain. Mukesh has eight years of experience in the information technology industry, most of which has been in the banking and financial services sector. Before joining ZenTEST Labs, Mukesh worked with Kanbay, Capgemini Consulting, and Fortune 500 clients, including Morgan Stanley and Merrill Lynch. He has played a major role in designing functional automation processes at the organizational level. Mukesh is a regular presenter at STARWEST and STAREAST.

**TP** **Cover Your Testing Bases with the Agile Testing Quadrants** **NEW***Lisa Crispin, ePlan Services, Inc.*

Software quality has many dimensions, so we must be ready with different testing approaches. We test to find defects, ensure system reliability, check that the system is easy to use, verify that it's secure, and much more. How do you know the different types of tests you need? How do you know when you're "done" testing? Lisa Crispin shows you how to use the four categories of the Agile Testing Quadrants method to make sure your team has covered all the bases—programmer tests (test-driven development); customer tests that help the team meet the users' requirements; business-facing tests that critique the product's behavior and find the important bugs; and technology-facing tests that examine non-functional qualities such as performance, load, scalability, reliability, and security. With Agile Testing Quadrants, you and your team will be ready to cover all the bases—no matter what testing challenges the project throws at you.



A tester on agile teams since 2000, **Lisa Crispin** currently works as a tester at ePlan Services, Inc. developing Web-based financial applications using XP and Scrum. She also helps teams and testers transition to agile, leads tutorials and workshops on agile testing at conferences in the US and Europe, and regularly contributes articles about agile testing to publications such as Better Software magazine, IEEE Software, and Methods & Tools. Lisa co-authored Testing Extreme Programming with Tip House and is co-authoring Agile Testing: The Tester Role on an Agile Project (2009) with Janet Gregory. For more about Lisa's work, visit her Web sites: <http://lisa.crispin.home.att.net> and <http://www.agiletester.ca>.

**TQ** **The Fundamentals of Pairwise Testing—With a Twist***Randall Rice, Rice Consulting Services, Inc.*

Pairwise testing is a technique for designing test cases to include all possible discrete combinations of each pair of input parameters. Use case scenarios define a dialogue between a user and the system with a tangible result. Join Randall Rice to explore ways to employ these two techniques in combination to design tests that provide a high level of test coverage while minimizing the total number of tests needed. Learn practical ways to prioritize pairwise test scenarios by risk level and execution effort as you gain a new tool to increase both coverage and efficiency in your testing. This optimized scenario-based approach is useful for establishing a strong baseline of regression tests that are both compact and achievable in most project schedules. In addition, pairwise scenario-based test designs used together are a powerful technique for system testing, user acceptance testing, and testing new service-oriented architectures (SOAs).



**Randall Rice** is a leading author, speaker, and consultant in the field of software testing and software quality. He has worked with major organizations worldwide to improve the quality of their information systems and to optimize their testing processes. Randall has more than thirty years of experience building and testing mission-critical projects in a variety of environments and has authored many training courses in software testing and software engineering. He is publisher of The Software Quality Advisor newsletter and co-author (with William E. Perry) of Surviving the Top Ten Challenges of Software Testing. Randall serves on the board of directors of the American Software Testing Qualifications Board (ASTQB).



Rob Sabourin

WEDNESDAY, OCTOBER 1, 8:45 a.m.

## Testing Lessons from Springfield—Home of the Simpsons

*Rob Sabourin, AmiBug.com*

Over the years, Rob Sabourin has discovered testing lessons in the Looney Tunes gang, The Great Detectives, and Dr. Seuss. Now he turns his attention to the Simpsons, a primetime cartoon television show entertaining audiences since 1989. Rob believes that Matt Groening's popular characters can teach us important lessons about software testing. Homer's twisted ideas tell us about test automation—why it works and why it fails. Could your software stand up to Bart's abuse? Lisa Simpson, the brilliant but neglected middle child, provides a calming influence on projects. Apu, the Kwik-E-Mart operator, works 100 hours a week—should you? When is Montgomery Burn's authoritarian management style effective? And can we bribe stakeholders as easily as Police Chief Wiggum takes a donut? Inside this simple cartoon are lessons on personas, context, organization, ethics, situational leadership, and motivation. Just like you, the people of Springfield commit to absurdly complex projects, such as the Monorail, all of which ultimately fail miserably. Join Rob in a revealing "Simpsons retrospective" loaded with tons of testing lessons from Springfield.

*Rob Sabourin has more than twenty-five years of management experience, leading teams of software development professionals. A well-respected member of the software engineering community, Rob has managed, trained, mentored, and coached hundreds of top professionals in the field. He frequently speaks at conferences and writes on software engineering, SQA, testing, management, and internationalization. The author of I am a Bug!, the popular software testing children's book, Rob is an adjunct professor of Software Engineering at McGill University.*



Jon Bach

WEDNESDAY, OCTOBER 1, 10:00 a.m.

## Telling Your Exploratory Story

*Jon Bach, Quardev, Inc.*

What do you say when your manager asks, "How did it go today?" As a test manager, you might say, "I'll check to see how many test cases the team executed today." As a tester with a pile of test cases on your desk, you could say, "I ran forty percent of these tests today," or "At the rate I'm going, I'll be finished with these test cases in forty days." However, if you're using exploration as part of your testing approach, it might be terrifying to try to give a status report—especially if some project stakeholders think exploratory testing is irresponsible and reckless compared to test cases. So how can you retain the power and freedom of exploration and still give a report that earns your team credibility, respect, and perhaps more autonomy? Jon Bach offers ways for you to explain the critical and creative thinking that makes exploratory testing so powerful. Learn how to report your exploration so stakeholders have a better understanding and appreciation of the value of exploratory testing to your project.

*Jon Bach is senior consultant and manager for corporate intellect at Quardev, Inc., a Seattle outsource test lab where he manages testing projects ranging from a few days to several months using Rapid Testing techniques. In 2000, Jon and his brother James invented the "Session-Based Test Management" technique for managing and measuring exploratory testing. In his thirteen years of testing, Jon has been a test contractor, full-time test manager, and consultant for companies such as Microsoft and Hewlett-Packard. He has written articles for both Better Software and IEEE Computer magazines.*



Julian Harty

WEDNESDAY, OCTOBER 1, 4:30 p.m.

## Six Thinking Hats for Software Testers

*Julian Harty, Google*

Our testing is only as good as our thinking—and all too often we are hampered by limiting ideas, poor communication, and pre-set roles and responsibilities. Based on the work of Edward de Bono, the Six Thinking Hats for software testers have helped Julian, and numerous others, work more effectively as testers and managers. The concepts are simple and easy to learn. For instance, we can use these concepts as individuals performing reviews and while testing, and in groups during team meetings. Each of the six hats has a color representing a direction of thinking—the blue hat provides the overview and helps to keep us productive; the white hat helps us to collect facts; the red hat is a way to express intuition and feelings without having to justify them; the yellow hat seeks the best possible outcome; the black hat helps us to discover what might go wrong—not only with the software but also with our tests and our assumptions! Finally, the green hat enables us to find creative solutions to ideas and issues discovered with the other five hats. Come and learn how to apply the six testing hats and other "thinking skills" on your test projects.

*A senior test engineer at Google, Julian Harty finds ways to test lots of fun products, including the mobile wireless software used by millions of users worldwide. He's been involved in software and online systems for more than twenty years and enjoys working with others to find ways to solve testing challenges productively. A presenter at both STAREAST and STARWEST, Julian has been involved in international conferences and workshops on software testing.*





Julie Gardiner

THURSDAY, OCTOBER 2, 8:30 a.m.

## Branch Out Using Classification Trees

*Julie Gardiner, Grove Consultants*

Classification trees are a structured, visual approach to identify and categorize equivalence class partitions for test objects. They enable testers to create better test cases faster. Classification trees visually document test requirements to make them easy to create and comprehend. Julie Gardiner explains this powerful technique and how it helps all stakeholders understand exactly what is involved in testing and offers an easier way to validate test designs. Using examples, Julie shows you how to create classification trees, how to construct test cases from them, and how they complement other testing techniques in every stage of testing. Julie demonstrates a free classification tree editing tool that helps you build, maintain, display, and use classification trees. Using the classification tree technique and tool, you keep test documentation to a minimum, more easily create and maintain regression tests, and drastically reduce test case bloat to make your test suites more usable.

With more than eighteen years of experience in the IT industry, **Julie Gardiner** has spent time as an analyst programmer, Oracle DBA, and project manager. She has first-hand experience as a test analyst, test team leader, test consultant, and test manager. At Grove Consultants, Julie provides consultancy and training in all aspects of testing, specializing in risk-based testing, agile testing, test management, and people issues. She is a certified ScrumMaster. Julie won best presentation at STAREAST 2007 and 2005; best presentation at BCS SIGIST 2005; and best tutorial at EuroSTAR 2006.



Gerard Meszaros

THURSDAY, OCTOBER 2, 4:15 p.m.

## Has the Time for the Adversarial Organization Passed?

*Gerard Meszaros, Independent Consultant*

The concept of an independent test organization is considered a “best practice” by many experts in the industry. Is this degree of autonomy actually a good thing in the real world today? In such a structure, some testers can only play “Battleship” with the delivered software, shouting gleefully when they find a defect. On their first tours of Toyota’s factories, American automakers were astonished to find no “rework area.” Toyota engineers didn’t subscribe to the approach of inserting defects on the production line only to remove them later in the quality control and rework area. Yet this is exactly what the independent test group excels at! Is it time to discard this organizational model and focus on working together with developers to prevent defects in the first place? Gerard Meszaros examines the sacred concept of independent test teams based on experiences from the agile software movement and Lean production systems. Both have shown that it is possible to replace the often dysfunctional, blaming relationship between the builders and the customers with one of mutual respect and cooperation. By applying the same “whole team” model within the technology organization, Gerard proposes to build quality in from the beginning rather than trying to test it in after the fact.

A Calgary, Canada-based consultant and trainer, **Gerard Meszaros** specializes in agile development processes. He has more than twenty-five years of experience building and testing software intensive systems in both product development and IT environments with technologies ranging from Java and .NET to Ruby and SAP’s ABAP. Gerard coaches cross-functional teams as they learn how to better envision, specify, develop, and test software systems using agile methods. He is a frequent speaker at major international software conferences and is the author of xUnit Test Patterns—Refactoring Test Code.



Tara Roth

FRIDAY, OCTOBER 3, 8:30 a.m.

## Testing Microsoft Office®: Experiences You Can Leverage to Drive Quality Upstream

*Tara Roth, Microsoft*

Have you experienced those weeks when the new features being added to builds just flat out don’t work? Do you strive to have a testable build throughout the full product development cycle? Are you tired of the mountain of bugs crushing you just before time to ship? Experienced test manager Tara Roth discusses how the Microsoft Office team is working to drive the level of test coverage up during the earlier phases of product development to improve build quality later in development. Tara describes two approaches, adopted by Microsoft Office, that improved efficiency and quality—Feature Crews and Big Button. Feature Crews is a tight-knit partnership of the developer, tester, and program manager who work together on a private release of new code prior to checking it in to the main build. Big Button is an approach to having the team kick off an automated suite of tests prior to checking in to the main build. Tara explains their successes and describes how you can apply these concepts in your organization. In addition to sharing her Microsoft Office experience, Tara describes how other Microsoft projects apply these techniques and how you can do the same.

The director of test for Microsoft Office, **Tara Roth** has sixteen years of professional experience in the software industry. Prior to her current role, she worked on Windows SharePoint Services®, Microsoft Project®, and Microsoft FrontPage®. In 2005, Tara and a few other test leaders at Microsoft, together with countless volunteers, pulled together “Test Day”—a day filled with engaging speakers and a tradeshow-style event—which has continued as an annual tradition at Microsoft. Tara is very passionate about testing and the important role we all play in software quality.

WEDNESDAY, OCTOBER 1, 11:30 a.m.

## W1 TEST MANAGEMENT

### The Myth of Risk Management

Pete McBreen, Software Craftsmanship, Inc.

Although test managers are tasked with helping manage project risks, risk management practices used on most software projects produce only an illusion of safety. Many software development risks cannot be managed because they are unknown, unquantifiable, uncontrollable, or unmentionable. Rather than planning only for risks that have previously occurred, project and test managers must begin with the assumption that something new will impact their project. The secret to effective risk management is to create mechanisms that provide for the early detection and quick response to such events—not simply to create checklists of problems you’ve previously seen. Pete McBreen presents risk “insurance” as a better alternative to classic risk management. He offers a risk insurance model, which helps insure projects against incomplete information, minor slippages that add up to major delays, late breaking bad news, and failure to learn from the past. Join Pete to learn how your testing projects can be flexible, responsive, and better able to deal with your project’s risks—both known and unknown.

## W2 TEST TECHNIQUES

### Using Failure Modes to Power Up Your Testing

Dawn Haynes, PerfTestPlus, Inc.

When a tester uncovers a defect, it usually gets fixed. The tester validates the fix and may add the test to a regression test suite. Often, both the test and defect are then forgotten. Not so fast—defects hold clues about where other defects may be hiding and often can help the team learn to not make the same mistake again. Dawn Haynes explores methods you can use to generate new test ideas and improve software reliability at the same time. Learn to use powerful analysis tools, including FMEA—failure modes and effects analysis—and cause/effect graphing. Go further with these techniques by employing fault injections and forensically analyzing bugs that customers find. Discover ways to correct the cause of a problem rather than submitting a “single instance defect” that will result in a “single instance patch” that fixes one problem and does nothing to prevent new ones. Learn how to power up your testing to reveal defect patterns and root causes for recurring defects.

## W3 TEST AUTOMATION

### Adventures with Test Monkeys

John Fodeh, Hewlett-Packard

Most test automation focuses on regression testing—repeating the same sequence of tests to reveal unexpected behavior. Despite its many advantages, this traditional test automation approach has limitations and often misses serious defects in the software. John Fodeh describes “test monkeys,” automated testing that employs random inputs to exercise the software under test. Unlike regression test suites, test monkeys explore the software in a new way each time a test case executes and offers the promise of finding new and different types of defects. The good news is that test monkey automation is easy to develop and maintain and can be used early in development before the software is stable. Join John to discover different approaches you can take to implement test monkeys, depending on the desired “intelligence” level. Learn to use weighted probability tables to direct your test monkeys into specific areas of interest, and find out how monkeys can work with model-based testing to make your testing even more powerful.

## W4 PERSONAL EXCELLENCE

### Five Things Every Tester Must Do

Julie Gardiner, Grove Consultants

Are you a frustrated tester or test manager? Are you questioning whether or not a career in testing is for you? Do you wonder why others in your organization seem unenthusiastic about quality? If the answer is yes to any of these questions, this session is for you. Julie Gardiner explores five directives to help testers make a positive impact within their organization and increase professionalism in testing. Remember quality—it’s not just time, it’s time and quality; it’s date and quality; it’s functionality and quality. Learn to enjoy testing and have fun—the closest job to yours is blowing up things for the movies. Relish the testing challenge—it’s you against the software and sometimes, it seems, the world. Choose your battles—take a stand on issues that are vital and let the small things go. And most importantly, remember that the only real power we have springs from our integrity—don’t sell that at any price. Join Julie for this important and inspirational session. You’ll be glad you did.

## W5 SPECIAL TOPICS

### Fun with Regulated Testing

John McConda, Mobius Test Labs

Does your test process need to pass regulatory audits (FDA, SOX, ISO, etc.)? Do you find that an endless queue of documentation and maintenance is choking your ability to do actual testing? Is your team losing good testers due to boredom? With the right methods and attitude, you can do interesting and valuable testing while passing a process audit with flying colors. It may be easier than you think to incorporate exploratory techniques, test automation, test management tools, and iterative test design into your regulated process. You’ll be able to find better bugs more quickly and keep those pesky auditors happy at the same time. John McConda shares how he uses exploratory testing with screen recording tools to produce the objective evidence auditors crave. He explains how to optimize your test management tools to preserve and confidently present accountability and traceability data. Learn to negotiate which test activities are auditable and create tests with an iterative test design approach that quickly adapts to change and makes auditors smile.

*"I like connecting with other people who do this thing we do and who are smart and passionate. I also realized that not only am I not alone, I'm part of a very thoughtful and curious group."*



Danny Vinson  
Software QA Manager,  
Kronos, Inc.

**W6 TEST MANAGEMENT****Great Test Teams Don't Just Happen***Jane Fraser, Electronic Arts*

Test teams are just groups of people who work on projects together. But how do great test teams become great? More importantly, how can you lead your team to greatness? Jane Fraser describes the changes she made after several people on her testing staff asked to move out of testing and into other groups—production and engineering—and how helping them has improved the whole team and made Jane a much better leader. Join Jane as she shares her team's journey toward greatness. She started by getting to really know the people on the team—what makes them tick, how they react to situations, what excites them, what makes them feel good and bad. She discovered the questions to ask and the behaviors to observe that will give you the insight you need to lead. Join Jane to learn how to empower your team members with the responsibility, authority, and accountability to get the job done while you concentrate on removing roadblocks to their success. And most importantly, remember it's all about them—it's not about you.

**W7 TEST TECHNIQUES****Understanding Test Coverage***Michael Bolton, DevelopSense*

Test coverage of application functionality is often poorly understood and always hard to measure. If they do it at all, many testers express coverage in terms of numbers, as a percentage or proportion—but a percentage of what? When we test, we develop two parallel stories. The “product story” is what we know and can infer about the software product—important information about how it works and how it might fail. The “testing story” is how we modeled the testing space, the oracles that we used, and the extent to which we configured, operated, observed, and evaluated the product. To understand test coverage, we must know what we did not test and that what we did test was good enough. Michael Bolton proposes alternatives for obtaining and describing test coverage—diagrams, strategy models, checklists, spreadsheets and matrices, and dashboards—and suggests how we can use these tools to build a clearer understanding of coverage to illuminate both the product story and the testing story.

**W8 TEST AUTOMATION****Automate API Tests with Windows PowerShell***Nikhil Bhandari, Intuit*

Although a myriad of testing tools have emerged over the years, only a few focus on the area of API testing for Windows-based applications. Nikhil Bhandari describes how to automate these types of software tests with Windows PowerShell, the free command line shell and scripting language. Unlike other scripting shells, PowerShell works with WMI, XML, ADO, COM, and .NET objects as well as data stores, such as the file system, registry, and certificates. With PowerShell, you can easily develop frameworks for testing—unit, functional, regression, performance, deployment, etc.—and integrate them into a single, consistent overall automation environment. With PowerShell, you can develop scripts to check logs, events, process status, registry check, file system management, and more. Use it to parse XML statements and other test files. Reduce your testing cycle times to better support iterative development and, at the same time, have more fun testing your Windows applications.

**W9 PERSONAL EXCELLENCE****What Price Truth? When a Tester is Asked to Lie***Fiona Charles, Quality Intelligence, Inc.*

As testers and test managers, our job is to tell the truth about the current state of the software on our projects. Unfortunately, in the high-stakes business of software development, often there is pressure—subtle or overt—to distort our messages. When projects are late or product reliability is poor, managers' and developers' reputations—and perhaps even their jobs—may be on the line. Fiona Charles discusses the importance to testers of refusing to compromise the truth, recognizing a potential cover-up before it occurs, knowing the legal position around securing project information, and developing a strategy to maintain integrity and still get out alive. She examines the early warning signs and discusses the practical tactics available to testers, including signaling your unwillingness to lie, getting accurate and detailed reports of project progress and status on the record, keeping notes of disturbing conversations and events, and choosing whether to “blow the whistle” (and if so, to whom), leave the organization—or both.

**W10 SPECIAL TOPICS****The Case Against Test Cases***James Bach, Satisfice*

A test case is a kind of container. You already know that counting the containers in a supermarket would tell you little about the value of the food they contain. So, why do we count test cases executed as a measure of testing's value? The impact and value a test case actually has varies greatly from one to the next. In many cases, the percentage of test cases passing or failing reveals nothing about the reliability or quality of the software under test. Managers and other non-testers love test cases because they provide the illusion of both control and value for money spent. However, that doesn't mean testers have to go along with the deceit. James Bach stopped managing testing using test cases long ago and switched to test activities, test sessions, risk areas, and coverage areas to measure the value of his testing. Join James as he explains how you can make the switch—and why you should.

*"This was my first  
STAR conference and it was  
fabulous. Wonderful to get new  
perspectives on topics and it really refreshed  
my attitude and confidence in my own career."*



Jen Tritch  
Software Quality Analyst,  
Catalina Marketing



WEDNESDAY, OCTOBER 1, 3:00 p.m.

## W11 TEST MANAGEMENT

### Test Estimation: Painful or Painless?

Lloyd Roden, Grove Consultants

As an experienced test manager, Lloyd Roden believes that test estimation is one of the most difficult aspects of test management. You must deal with many unknowns, including dependencies on development activities and the variable quality of the software you test. Lloyd presents seven proven ways he has used to estimate test effort. Some are easy and quick but prone to abuse; others are more detailed and complex but may be more accurate. Lloyd discusses FIA (finger in the air), formula/percentage, historical reference, Parkinson's Law vs. pricing, work breakdown structures, estimation models, and assessment estimation. He shares spreadsheet templates and utilities that you can use and take back to help you improve your estimations. By the end of this session, you might just be thinking that the once painful experience of test estimation can, in fact, be painless. *Useful utilities will be given out during the session to help with estimation.*

## W12 TEST TECHNIQUES

### Exploratory Testing: The Next Generation

David Gorena Elizondo, Microsoft

Exploratory testing is sometimes associated with "ad hoc" testing, randomly navigating through an application. However, emerging exploratory techniques are anything but ad hoc. David Gorena Elizondo describes new approaches to exploratory testing that are highly effective, very efficient, and supported by automation. David describes the information testers need for exploration, explains how to gather that information, and shows you how to use it to find more bugs and find them faster. He demonstrates a faster and directed (not accidental) exploratory bug finding methodology and compares it to more commonly used approaches. Learn how test history and prior test cases guide exploratory testers; how to use data types, value ranges, and other code summary information to populate test cases; how to optimize record and playback tools during exploratory testing; and how exploratory testing can impact churn, coverage, and other metrics.

## W13 TEST AUTOMATION

### Test Automation Techniques for Dynamic and Data Intensive Systems

Chris Condron, The Hanover Group

If you think you're doing everything right with test automation but it just won't scale, join the crowd. If the amount of data you're managing and the dynamic changes in applications and workflows keep you in constant maintenance mode, this is the session for you. Encountering these problems, Chris Condron's group reviewed their existing automation successes and pain points. Based on this analysis, they created a tool agnostic architecture and automation process that allowed them to scale up their automation to include many more tests. By aligning their test scripts with the business processes, his team developed a single test case model they use for both manual and automated tests. They developed a test data management system incorporating storage of and a review process for three types of test data: scenarios, screen mappings, and references. Their new test scripts contain only the application flow information and reference the test data system for the input values. Join Chris and find out how you can enjoy the same success.

## W14 PERSONAL EXCELLENCE

### Top Ten Non-Technical Skills for Better Testing

Krishna Iyer and Mukesh Mulchandani, ZenTEST Labs

In the era of SOA and Web 2.0, as it becomes more and more difficult to accomplish comprehensive testing, Krishna Iyer and Mukesh Mulchandani describe ten non-technical skills that will make you a better tester. The first five are qualities we often look for in testers yet seldom practice scientifically and diligently—collaboration, creativity, experimentation, passion, and alertness. The second five are abilities that are seldom mentioned, yet equally important for testers—connect the dots, challenge the orthodox, picture and predict, prioritize, and leave work at work. Drawing from their experiences of building a testing team for their organization and consulting with global firms in building "testing capability," Krishna and Mukesh show how you and your test team can improve each of these ten non-technical skills. Practice these skills during the session and take back techniques you can use to hone your skills at work.

## W15 SPECIAL TOPICS

### The Power of Specially Gifted Software Testing

Thorkil Sonne, Specialisterne

Specialisterne ("The Specialists") is a Danish company that employs people with very special capabilities to perform complex and difficult tasks, including software testing, quality control, and data conversion. Their customers are companies such as Computer Sciences Corporation (CSC), Microsoft, and leading Danish IT organizations. Their founder and our presenter, Thorkil Sonne, received the IT Award 2008 from the Danish IT Industry Association for the company's ability to find and employ especially talented people in IT. Seventy-five percent of the employees of Specialisterne have autism—Autistic Spectrum Disorder (ASD)—typically Asperger's Syndrome. Traditionally, society has viewed people with ASD as handicapped. Yet, their abilities to concentrate, stick to tasks, and quickly absorb highly complex technical information are exactly the characteristics of the best software testers. Hear Thorkil's vision and strategy on how he believes the software testing industry worldwide can derive considerable benefit from employing special people with autism.

*"So many choices! Loved that I can learn so much about things I don't currently do on my job. Like Lee said, if you don't do it, who will? Wonderful event! I loved it."*



Lynn Russell  
Software Test Engineer,  
Follett Software Company

**T1 TEST MANAGEMENT****Quality Metrics for Testers: Evaluating Our Products, Evaluating Ourselves***Lee Copeland, Software Quality Engineering*

As testers, we focus our efforts on measuring the quality of our organization's products. We count defects and list them by severity; we compute defect density; we examine the changes in those metrics over time for trends, and we chart customer satisfaction. While these are important, Lee Copeland suggests that to reach a higher level of testing maturity, we must apply similar measurements to ourselves. He suggests you count the number of defects in your own test cases and the length of time needed to find and fix them; compute test coverage—the measure of how much of the software you have actually exercised under test conditions—and determine Defect Removal Effectiveness—the ratio of the number of defects you actually found divided by the total number you should have found. These and other metrics will help you evaluate and then improve the effectiveness and efficiency of your testing process.

**T2 TEST TECHNIQUES****Patterns and Practices for Model-Based Testing***Keith Stobie, Microsoft*

To apply model-based testing (MBT) to many different applications, simply learning the high-level principles is not enough. You need extra guidance and practice to help orient testers and developers to begin using models for testing. Many people attempting MBT, confused about programming around observation and control, try to duplicate the underlying system functionality in models. Keith Stobie shows you real-world MBT case studies to illustrate MBT ideas you can incorporate into your own practices. Learn to apply MBT patterns and practices to both traditional and model-based test design. See abstracting examples and how these abstractions can help testers with any test suite—model-based or not. Learn to create adapters that act as a specialized programming language—similar to keyword-based testing—for the abstractions of your domain under test. Detect under-testing and over-testing by creating a logging framework using assertions to trace tests back to requirements. With MBT patterns and practices, you can do MBT—More Better Testing!

**T3 TEST AUTOMATION****End-to-End Test Automation for Complex Systems***Thomas Thunell, Ericsson AB*

As a world-leading provider of telecommunications equipment, Ericsson knows that test automation is a key factor for driving a successful test organization. Thomas Thunell describes their automation solution—a test system for complex, end-to-end environments. Ericsson's networks typically consist of mobile terminals, base stations, radio network controllers, switching systems, protocol analyzers, and possibly other components. Thomas discusses the lessons Ericsson has learned—obtain management commitment up front, use dedicated automation teams, and take the long-term view in automation work. When it came to planning, establishing guidelines, and getting the right people on board, Ericsson treated test automation exactly the same as any other software development project. In so doing, they built—and depend on—a rock-solid, easy-to-use, reliable test automation framework. Future plans include automated post-processing of test logs and delivering test automation metrics directly from the system. Find out how Ericsson is doing test automation to see how you can follow their path.

**T4 THE NEW WAVE****Testing AJAX Applications with Open Source Tools***Frank Cohen, PushToTest*

AJAX testers and developers have serious challenges developing unit tests, functional tests, and load/performance tests in a time when AJAX and other Web development technologies continue to expand. Frank Cohen explains a proven methodology to identify—and solve—scalability, performance, and reliability issues in AJAX applications. Frank explains how to apply this methodology using open source testing tools, including Selenium, soapUI, TestGen4Web, PushToTest, and others. He demonstrates hands-on testing examples created with the Appcelerator and Google Widget Toolkit (GWT) frameworks. You'll also see how to construct a functional unit test for a business flow, identify ways to create operational test data at run time, validate test responses, and automate the entire test. Learn to use Firebug and Firefox to identify and instrument AJAX user interface elements. Add these new tools and methods to your toolkit for a better AJAX testing experience.

**T5 SPECIAL TOPICS****Man and Machine: Combining Tools with the Human Mind***Jonathan Kohl, Kohl Concepts, Inc.*

When you think of automated testing, you usually think of computer software executing unattended tests. When you think of manual testing, you think of a human being executing tests without the aid of software. Instead of thinking of tests as either automated or manual, Jonathan Kohl explores ways you can blend the two. He helps you answer the questions, "How can automation improve my exploratory and scripted testing work?" and "What do we lose if we run these tests without any human supervision?" With numerous examples, Jonathan demonstrates the different mindset he uses to implement test automation as he highlights techniques from a hybrid testing approach. He demonstrates examples from his personal testing experiences and from other disciplines to change your mind on man and machine testing.

*"A lot of really great information packed into a few days. Great use of humor, and comparison of other fields to software testing. My first conference, I will definitely come again if I am able. Also hope other testers from my company can attend. Thanks!"*

**Business Analyst Tester**

THURSDAY, OCTOBER 2, 11:15 a.m.

## T6 TEST MANAGEMENT

### Calculate the Value of Testing: It's Not Just About Cost

*Leo van der Aalst, Sogeti Netherlands BV*

It seems that senior management is always complaining that testing costs too much. And their opinion is accurate if they consider only the costs—and not the benefits—of testing. What if you could show management how much you have saved the organization by finding defects during testing? The most expensive defects are ones not found during testing—defects that ultimately get delivered to the user. Their consequential damages and repair costs can far exceed the cost of finding them before deploying a system. Instead of focusing only on the cost of testing, Leo van der Aalst shows you how to determine the real value that testing adds to the project. He shares a model that he has used to calculate the losses testing prevents—losses that did not occur because testing found the error before the application was put into production. Leo explains the new testing math: Loss Prevented – Cost of Testing = Added Value of Testing.

## T7 TEST TECHNIQUES

### The Savvy Web Tester's Tool Kit

*Erik Petersen, emprove*

Did you know that you can get many free—or nearly free—tools to supercharge your Web testing efforts? Amazingly, at the click of a button, you can download some very advanced capabilities to make you seem like a testing genius. With a focus on Web application tools, Erik Petersen looks at tools that can help all testers. Erik examines mind mapping and how you can use mind maps for schedules, strategies, even tests themselves. He demonstrates several tools for managing tests and others to help you look “under the hood” and manipulate Web applications. Join Erik to learn some innovative ways to test your Web applications; build test data to include dummy people with realistic addresses; capture what you’ve done; and view, tweak, and break the software. You’ll also see “portable applications”, versions of tools that run off a memory stick on any PC without being installed.

## T8 TEST AUTOMATION

### Demystifying Virtual Test Lab Management

*Ian Knox, Skytap, Inc.*

The benefits of a virtualized test lab environment are compelling and quantifiable—rapid provisioning and tear down of environments, faster test cycles, and powerful new capabilities to resolve defects. Although many test teams have experimented with virtual machines and have experienced some of the benefits, they’ve also discovered issues with virtual machine “sprawl,” difficulties administering the lab, and lack of virtual private networking. Ian Knox provides solutions to these problems and offers ways to simplify both using and managing virtualization in your test environment. Ian describes the basics of virtualization and how you can use virtual labs to solve some of the most pressing and expensive challenges in testing. He guides you through the important implementation choices for building a virtual lab and explores the common pitfalls with real-life case studies. Take back an understanding of a virtual lab’s capabilities and limitations and learn how to automate your lab with tools and build integration.

## T9 THE NEW WAVE

### Building an SOA Quality Center of Excellence

*Rajeev Gupta, iTKO LISA*

Before we can realize the promises of technical agility and reuse from a distributed, service-oriented architecture (SOA), we must first establish trust among stakeholders that SOA will meet business requirements. Rajeev Gupta believes that the best way to instill this sense of trust and make SOA adoption possible is through a shared center of excellence focused on SOA quality. Both service providers and businesses consuming services must be confident that services and the underlying implementation and data layers behind them reliably meet business goals, even as they change and evolve over time. An SOA Quality Center of Excellence demonstrates that quality is everyone’s duty—not just the testing team’s responsibility. Learn the four key activities that the SOA Quality Center of Excellence must manage: structural certification, behavioral validation, performance testing, and service virtualization of test environments. If all stakeholders work together to ensure quality with a continuing focus, SOA can and will succeed in your organization.

## T10 SPECIAL TOPICS

### Beyond Functional Testing: On to Conformance and Interoperability

*Derk-Jan De Grood, Collis*

Although less well known than security and usability testing, conformance and interoperability testing are just as important. Even though conformance and interoperability testing—all about standards and thick technical specifications documents—may seem dull, Derk-Jan De Grood believes that these testing objectives can be interesting and rewarding if you approach them the right way. SOA is one example in which numerous services must interact correctly with one another—conform to specs—to implement a system. Conformance and interoperability testing ensures that vendors’ scanners can read your badge in the EXPO and that your bank card works in a foreign ATM. Derk-Jan explains important concepts of interface standards and specifications and discusses the varied test environments you need for this type of testing. Get insight into the problems you must overcome when you perform conformance and interoperability testing.



**See what a STAR experience can do for you and your team!**



**T11 TEST MANAGEMENT****Managing Your Personal Stress Level***Randall Rice, Rice Consulting Services, Inc.*

In a recent survey of 130 U.S. software testers and test managers, Randall Rice learned that 83 percent of the respondents have experienced burnout, 53 percent have experienced depression of some type, and 97 percent have experienced high levels of stress at some time during their software testing careers. Randall details the sources of these problems and the most common ways to deal with them—some healthy, some not. There are positive things testers and managers can do to reduce and relieve their stress without compromising team effectiveness. By understanding the proper role of testing inside your organization and building a personal support system, you can manage stress and avoid its destructive consequences. Randall identifies the stress factors you can personally alleviate and helps you deal with those stressors you can't change. Avoid burnout and don't be taken down by unreasonable management expectations, negative attitudes of other people, unexpected changes, and other stressors in your work.

**T12 TEST TECHNIQUES****Reloadable Test Data for Manual Testing***Tanya Dumaresq, Macadamian Technologies, Inc.*

Do you need to execute and then quickly re-execute manual test cases under tight timelines? Do bugs marked as "Cannot Reproduce" bouncing back and forth between developers and testers frustrate your team? Would you like to have more realistic, production-like test data? Join Tanya Dumaresq as she explains the hows and whys of developing and using pre-created, reloadable test data for manual testing. By planning ahead when designing test cases, you can cut test execution time in half and virtually eliminate those "works on my machine" bugs. Learn how to create and load test data in different formats and choose the one that is best for your application under test. Sometimes, you can even use the application itself to create the data! You'll end up with test data and an environment far more representative of your users' world than if you create data on the fly during test execution.

**T13 AGILE TESTING****Driving Development with Tests: ATDD and TDD***Elisabeth Hendrickson, Quality Tree Software, Inc.*

A perennial wish of testers is to participate early in the projects we test—as early as when the requirements are being developed. We also often wish for developers to do a better job unit testing their programs. Now with agile development practices, both of these wishes can come true. Development teams practicing acceptance test-driven development (ATDD) define system-level tests during requirements elicitation. These tests clarify requirements, uncover hidden assumptions, and confirm that everyone has the same understanding of what "done" means. ATDD tests become executable requirements that provide ongoing feedback about how well the emerging system meets expectations. Agile developers who also are practicing test-driven development (TDD) design methods create automated unit tests before writing component code. The result of ATDD + TDD is an automated set of system- and unit-level regression tests that execute every time the software changes. In this session, Elisabeth explains how ATDD and TDD work and demonstrates them by completely implementing a new feature in a sample application.

**T14 PERFORMANCE TESTING****Performance Engineering: More Than Just Load Testing***Rex Black, QA Software Consultant/Trainer*

Performance testing that is done once or a few times as part of the system test is not the right approach for many systems that must change and grow for years. Rex Black discusses a different approach—performance engineering—that is far more than performing load testing during the system test. Performance engineering takes a broad look at the environment, platforms, and development processes and how they affect a system's ability to perform at different load levels on different hardware and networks. While load testers run a test before product launch to alleviate performance concerns, performance engineers have a plan for conducting a series of performance tests throughout the development lifecycle and after deployment. A comprehensive performance methodology includes performance modeling, unit performance tests, infrastructure tuning, benchmark testing, code profiling, system validation testing, and production support. Find out the what, when, who, and how to conduct each of these performance engineering activities. As a performance engineer, you'll learn the questions you need to ask—early in the project—to identify risks for load, stress, capacity, and reliability.

**T15 SPECIAL TOPICS****Test Management for Very Large Programs: A Survival Kit***Graham Thomas, Independent Consultant*

In large organizations with multiple, simultaneous, and related projects, how do you coordinate testing efforts for better utilization and higher quality? Some organizations have opened Program Test Management offices to oversee the multiple streams of testing projects and activities, each with its own test manager. Should the Program Test Manager be an über-manager in control of everything, or is this office more of an aggregation and reporting function? Graham Thomas examines the spectrum of possible duties and powers of this position. He also shares the critical factors for successful program test management, including oversight of the testing products and deliverables; matrix management of test managers; stakeholder, milestone, resource, and dependency management; and the softer but vital skills of influence and negotiation with very senior managers. Relating experience gained on several large testing programs, Graham shares a practical model—covering the key test management areas of organization, people, process, tools, and metrics—that your organization can adapt for its needs.

*"Excellent learning experience.  
Get the latest developments in a comfortable  
environment in one of the most pleasant locations  
in the world. Overall, a great experience."*



Jan J. Nathan  
Test CAT Leader,  
Nato Programming Centre

THURSDAY, OCTOBER 2, 3:00 p.m.

## T16 TEST MANAGEMENT

### The Three Faces of Quality: Control, Assurance, Analysis

Stephen Michaud, Luxoft Canada

Many of the misunderstandings within software development organizations can trace their roots to different interpretations of the role of testers. The terms quality control (QC), quality assurance (QA), and quality analysis are often used interchangeably. However, they are quite different and require different approaches and very different skill sets. Quality control is a measurement of the product at delivery compared to a benchmark standard, at which point the decision is made to ship or reject the product. Quality assurance is the systematic lifecycle effort to assure that a product meets expectations in all aspects of its development. It includes processes, procedures, guidelines, and tools that lead to quality in each phase. Quality analysis evaluates historical trends and assesses the future customer needs as well as trends in technology to provide guidance for future system development. Stephen Michaud describes how to set yourself up in all three roles and covers the skills you need to be successful in each role.

## T17 TEST TECHNIQUES

### Acceptable Acceptance Testing

Grigori Melnik, Microsoft Corporation and Jon Bach, Quardev, Inc.

This is the tale of a team of software professionals at Microsoft patterns & practices group who wrote a book on software acceptance testing. Grigori Melnik was the content owner, writer, and project manager. Jon Bach was the writer, material producer, and the acceptance testing reality checker, ensuring that the project team used its own methods so the book would be acceptable to you, the reader. To develop the book, Grigori and Jon employed key ideas of agile projects—creating a backlog using story cards, working in short iterations, exploring requirements and expectations, building customer trust through iterative acceptance, and staying connected to the customer community through frequent preview releases, surveys, and interviews. They created a heuristic acceptance testing model for knowing when they had reached enough “acceptability” to stop “developing” the book and publish it. Join Grigori and Jon to discover how you can apply an innovative acceptance testing methodology to your software testing. You’ll learn how to implement an iterative and incremental acceptance testing approach on your next testing project.

## T18 AGILE TESTING

### Are Agile Testers Different?

Lisa Crispin, ePlan Services, Inc.

On an agile team everyone tests, blurring the lines between the roles of professional developers and testers. What’s so special about becoming an agile test professional? Do you need different skills than testers on traditional projects? What guides you in your daily activities? Lisa Crispin presents her “Top Ten” list of principles that define an agile tester. She explains that when it comes to agile testers, skills are important but attitude is everything. Learn how agile testers acquire the results-oriented, customer-focused, collaborative, and creative mindset that makes them successful in an agile development environment. Agile testers apply different values and principles—feedback, communication, simplicity, continuous improvement, and responsiveness—to add value in a unique way. If you’re a tester looking for your place in the agile world or a manager looking for agile testers, Lisa can help.

## T19 PERFORMANCE TESTING

### Life as a Performance Tester

Scott Barber and Dawn Haynes, PerfTestPlus, Inc.

At the core of most performance testing challenges and failed performance testing projects are serious misunderstandings and miscommunications within the project team. Scott Barber and Dawn Haynes share approaches to overcoming some of the most common frustrations facing performance testers today. Rather than simply telling you how to improve understanding and communicate performance testing concepts, Scott and Dawn demonstrate their approaches through an amusing role play of interactions between a lead performance tester and a non-technical executive. Based on real-life experiences (with names and places changed to protect the innocent, of course), they demonstrate ways for you to address questions such as, “Should we be doing performance, load, stress, or capacity testing?”, “How relevant and realistic (or not) is this load test?”, “How will we know if we are done?”, and “What is a concurrent user, anyway?” As you enjoy the interplay, you’ll learn valuable lessons that are sure to make your performance testing better and personally more rewarding.

## T20 SPECIAL TOPICS

### Adding Measurement to Reviews

Riley Rice, Booz Allen Hamilton

Conceptually, most testers and developers agree that reviews and inspections of software designs and code can improve software and reduce development costs. However, most are unaware that measuring reviews and inspections greatly magnifies these improvements and savings. Riley Rice presents data from more than 4,000 real-world software projects in different domains—defense, commercial, and government. He compares the results of three scenarios: doing few or no reviews, doing unmeasured reviews, and doing measured reviews. For each scenario, Riley compares resulting metrics: defects delivered to customers, total project pre-release costs, total project post-release costs, total project lifecycle costs, project duration, mean time between failures, and productivity. The results are surprising—measured reviews are substantially more effective—and go far beyond what most people would expect. Learn how the effectiveness of inspections and reviews is significantly improved by the simple act of measuring them.

*"The conference is definitely a great place to come to meet the SQA world and get lots of different information and ideas. I traveled 9 hours to get to this conference and I will do it again."*



Valérie Simons-Leblay  
SQA Manager, IHS

**F1 TEST MANAGEMENT****Toward an Exploratory Testing Culture***Rob Sabourin, AmiBug.com, Inc.*

Traditional testing teams often agonize over exploratory testing. How can they plan and design tests without detailed up-front documentation? Stubborn testers may want to quit because they are being asked to move out of their comfort zone. Can a team's testing culture be changed? Rob Sabourin describes how several teams have undergone dramatic shifts to embrace exploratory testing. Learn how to blend cognitive thinking skills, subject matter expertise, and "hard earned" experience to help refocus your team and improve your outcomes. Learn to separate bureaucracy from thinking and paperwork from value. Explore motivations for change and resistance to it in different project contexts. Leverage Parkinson's Law—*work expands to fill the time available*—and Dijkstra's Principle—*testing can show the presence of bugs, but not their absence*—to inspire and motivate you and your team to get comfortable in the world of exploratory testing.

**F2 TEST TECHNIQUES****Truths and Myths of Static Analysis***Paul Anderson, GrammaTech*

Identifying defects with static analysis tools has advanced significantly in the last few years. Yet, there still are many misconceptions about the capabilities and limits of these innovative tools—and sales propaganda such as "100% path coverage" has not helped at all. Paul Anderson debunks common myths and clarifies the strengths and limitations of static-analysis technology. You'll learn about the types of defects that these tools can catch and the types they miss. Paul demystifies static analysis jargon, explaining terms such as object-sensitive and context-sensitive. Find out how the FDA uses static analysis today to evaluate medical device software. Paul jump-starts your understanding of static analysis so you can decide where to apply this technology and have more knowledge and confidence in your interactions with tool vendors.

**F3 AGILE TESTING****Lessons Learned in Acceptance Test-Driven Development***Antony Marciano, testingReflections.com*

Acceptance Test-Driven Development (ATDD), an application of the test-first practice of XP and agile development, can add enormous value to agile teams that are proficient in these practices. Moving from awareness of ATDD to being proficient at practicing ATDD comes about only after learning some important lessons. First, no one group can "own" the process. Second, ATDD is first about helping the customer and the team understand the problem; then it is about testing. Third, writing automated acceptance tests in ATDD is not the same as writing automated tests with typical automation tools. Antony Marciano shares his experiences with ATDD—the good, the bad, and the ugly—and the many other lessons he's learned in the process. Discover the benefits and pitfalls of ATDD and take advantage of Antony's experiences so that you avoid common mistakes that teams make on their journey to becoming proficient practitioners of ATDD.

**F4 SECURITY****Automating Security Testing with cUrl and Perl***Paco Hope, Cigital*

Although all teams want to test their applications for security, our plates are already full with functional tests. What if we could automate those security tests? Fortunately, most Web-based and desktop applications submit readily to automated testing. Paco Hope explores two flexible, powerful, and totally free tools that can help to automate security tests. cUrl is a free program that issues automatic basic Web requests; Perl is a well-known programming language ideally suited for writing test scripts. Paco demonstrates the basics of automating tests using both tools and then explores some of the more complicated concerns that arise during automation—authentication, session state, and parsing responses. He then illustrates simulated malicious inputs and the resulting outputs that show whether the software has embedded security problems. The techniques demonstrated in this session apply equally well to all Web platforms and all desktop operating systems. You'll leave with an understanding of the basics and a long list of resources you can reference to learn more about Web security test automation.

**F5 SPECIAL TOPICS****Database Locking: What Testers Should Know, Why Testers Should Care***Justin Callison, Luxoft Canada*

Database locking is a complicated technical issue for some testers. Although we often think that this issue belongs in the realm of the developer and the DBA—"It's not my problem"—database locking is the enemy of functional and performance testers. As Justin Callison can personally attest, locking defects have led to many disasters in production systems. However, there is hope! Justin sheds light on the problem of database locking, how it varies among different platforms, and the application issues that can arise. Armed with a new understanding of database locking, you can develop effective testing strategies. Join in and learn about these strategies: designing explicit locking tests, ensuring appropriate test data, implementing sufficient monitoring, and combining manual with automated testing to avoid disaster.

*"Variety of speakers—some with very different viewpoints. Great that it's not homogenized to one view. Amazing breadth of coverage—thanks! SQE Rocks!"*



**Jim Peak**  
Tester, Future Point Systems



FRIDAY, OCTOBER 3, 11:15 a.m.

## F6 TEST MANAGEMENT

### Keys to Successful Test Outsourcing: Win-Win for All

Patricia Smith, The Hartford

Defining and implementing a test outsourcing strategy can be a daunting task, especially if you have a vested interest in the outcome. Patricia Smith shares The Hartford's strategy for embarking on a vendor partnership for testing. She discusses three phases of successful outsourcing—preparing, determining benefits, and measuring—and shares tips on how to prepare to outsource before committing. Find out how to make the necessary critical decisions, including: selecting the right vendor, deciding what work “goes” and what “stays,” determining the right blend of employees and outsourced staff, and navigating the organizational infrastructure to support the team. Patricia explores the benefits of sourcing they have experienced—access to a huge labor pool and the “best” best practices. Take back ideas for measuring quality and implementing service level agreements (SLAs) that are simple and easy to track.

## F7 TEST TECHNIQUES

### A Modeling Framework for Scenario-Based Testing

Fiona Charles, Quality Intelligence, Inc.

Scenario-based testing is a powerful method for finding problems that really matter to users and other stakeholders. By including scenario tests representing actual sequences of transactions and events, you can uncover the hidden bugs often missed by other functional testing. Designing scenarios requires you to use your imagination to create narratives that play out through systems from various points of view. Basing scenarios on a structured analysis of the data provides a solid foundation for a scenario model. Good scenario design demands that you combine details of business process, data flows—including their frequency and variations—and clear data entry and verification points. Fiona Charles describes a framework for modeling scenario-based tests and designing structured scenarios according to these principles. Fiona works through a real-life project example, showing how she applied this framework to design tests that found hundreds of bugs in a system—and this after the company had completed their testing and delivered the system into acceptance.

## F8 AGILE TESTING

### Agile Acceptance Testing Using .NET FitNesse

Gojko Adzic, Neuri Ltd.

FitNesse is an open-source test automation tool that enables business users, developers, and testers to cooperate on agile acceptance testing. FitNesse allows them to build a shared understanding of system requirements that ultimately produces the software that is genuinely fit for its purpose. Gojko Adzic presents an introduction to agile acceptance testing. He discusses when to use FitNesse, when not to use it, and how to start writing acceptance tests with this free tool. Gojko explains how to make the most of automated acceptance tests by focusing on business rules, how to overcome workflow constraints, and how to avoid common testing pitfalls. He describes features specific to the .NET FitNesse test runner, including cell handlers and embedded symbols, that allow you to save time and effort in writing and maintaining tests. Join in to see if FitNesse fits into your .NET testing world.

## F9 SECURITY

### Integrating Security Testing into Your Process

Danny Allan, IBM Rational

Software quality is a priority for most organizations, yet many are still struggling to handle the volume of testing. Unfortunately, applications are frequently released with significant security risks. Many organizations rely on an overburdened security team to test applications late in development when fixes are the most costly, while others are throwing complex tools at test teams expecting the testers to master security testing with no formal processes and training. Danny Allan describes five steps to integrate security testing into the software development lifecycle. Danny shows how highly secure and compliant software applications begin with security requirements and include design, development, build, quality assurance, and transitional practices. He describes some of the most common application security vulnerabilities, techniques to address these issues, and methods to safeguard sensitive online information from the bad guys.

## F10 SPECIAL TOPICS

### Going Mobile: The New Challenges for Testers

Wayne Hom, Augmentum

Mobile device manufacturers face many challenges bringing quality products to market. Most testing methodologies were created for data processing, client/server, and Web products. As such, they often fail to address key areas of interest to mobile applications—usability, security, and stability. Wayne Hom discusses approaches you can use to transform requirements into usability guides and use cases into test cases to ensure maximum test coverage. He discusses automation frameworks that support multiple platforms to reduce test cycle times and increase test coverage, while measuring and reporting at the different phases of the software lifecycle. Wayne presents case studies to illustrate how to reduce test cycles by up to 75 percent. He demonstrates solutions that have helped providers of third party applications and services manage testing cycles for multiple mobile device releases.



## CONFERENCE BONUS!

One-Year Subscription to Better Software Magazine!

STARWEST 2008 conference attendees receive a one-year digital subscription (ten issues)

to Better Software magazine—the only magazine delivering relevant, timely information so you can tackle the challenges of building better quality software, regardless of your role in software development. [www.BetterSoftware.com](http://www.BetterSoftware.com)

If you are a current subscriber, your subscription will be extended for an additional ten digital issues.

# STARWEST 2008 REGISTRATION INFORMATION

SEPTEMBER 29–OCTOBER 3, 2008 ANAHEIM, CALIFORNIA, USA

## Easy to Register



### ONLINE:

[www.sqe.com/swreg](http://www.sqe.com/swreg)



### PHONE:

888.268.8770  
904.278.0524



### EMAIL:

[sqeinfo@sqe.com](mailto:sqeinfo@sqe.com)

## CONFERENCE PRICING

### Registration Fees:\*

	On or Before August 29	After August 29	
<input type="checkbox"/> Conference + 2 Pre-conference Tutorials	<b>\$2,195</b>	<b>\$2,395</b>	◀ <b>Best Value!</b>
<input type="checkbox"/> Conference + 1 Pre-conference Tutorial	\$1,995	\$2,195	
<input type="checkbox"/> Conference Only (Wed.-Fri.)	\$1,695	\$1,895	
<input type="checkbox"/> Pre-conference Tutorial (1 Day)	\$795	\$895	
<input type="checkbox"/> Pre-conference Tutorials (2 Days)	\$1,390	\$1,590	
<input type="checkbox"/> Certification Training Course + Conference**	\$3,740	\$3,940	◀ <b>A Savings of \$200!</b>
<input type="checkbox"/> Certification Training Course**	\$2,245	\$2,245	

### Special Early Bird Offer!

Receive \$200 off the regular conference registration fee if payment is received on or before August 29, 2008. See discounted pricing information above.

### PAYMENT INFORMATION

The following forms of payment are accepted: Visa, MasterCard, American Express, check, or company purchase order. Payment must be received before the registration is confirmed. Make all checks payable to Software Quality Engineering. You will receive a confirmation packet upon payment by check, credit card, or company purchase order. Payment must be received at Software Quality Engineering on or before August 29, 2008, to take advantage of the Early Bird conference rates listed above.

### HOTEL RESERVATIONS

Take advantage of the discounted conference rate at the *Disneyland®* Hotel in Anaheim, CA. To make a reservation, visit [www.sqe.com/go?SW08hotel](http://www.sqe.com/go?SW08hotel), or call 714.520.5005 and mention you are attending the STARWEST conference to receive your discount. Cancellations on a guaranteed reservation must occur more than 72 hours prior to the specified arrival time to ensure a refund. If you need special facilities or services, please notify the agent at the time of reservation. Make your reservation early.

### CANCELLATION POLICY

Conference registrations cancelled after September 8, 2008, are subject to a 20% cancellation fee. No cancellations or refunds may be made after September 15, 2008. Substitutions may be made at any time before the first day of the program. Call our Client Support Group at 888.268.8770 or 904.278.0524 to obtain a cancellation code. All valid cancellations require a cancellation code.

### SATISFACTION GUARANTEE

Software Quality Engineering is proud to offer a 100% satisfaction guarantee. If we are unable to satisfy you, we will gladly refund your registration fee in full.

### MEDIA RELEASE

From time to time we use photographs, audio, and video of conference participants in our promotional materials. By virtue of your attendance at STARWEST, you acknowledge that Software Quality Engineering, Inc., reserves the right to use your likeness in such materials.

\* Your registration fee includes \$39 for a one-year digital subscription (10 issues) to *Better Software* magazine. If you are a current subscriber, your subscription will be extended an additional ten digital issues.

\*\* A \$250 exam fee is included in the cost of the course registration.



## EVENT LOCATION

STARWEST will be held at the *Disneyland®* Hotel in Anaheim, California. The *Disneyland®* Hotel is the centerpiece of the *Disneyland®* Resort. A perfect balance of business and leisure facilities, the hotel features spacious, Disney-themed rooms; lots of shopping; Never Land Island, complete with white-sand beach and swimming pools; the Team Mickey Fitness Center; and themed restaurants. Within walking distance of lots of wonderful amenities, this facility is proof that you can mix business with pleasure.

### SPECIAL HOTEL RATES FOR STARWEST ATTENDEES!

Book your reservation for your stay at the *Disneyland®* Hotel at the discounted conference rate. If you need special facilities or services, please notify the agent at the time of reservation. To make a reservation, visit [www.sqe.com/go?SW08hotel](http://www.sqe.com/go?SW08hotel) or call 714.520.5005, by August 29, and mention you are attending the STARWEST conference to receive your discount. Cancellations on a guaranteed reservation must occur more than 72 hours prior to the specified arrival time to ensure a refund. Make your reservation early.

### ONLINE ACCESS AT THE CONFERENCE

All guestrooms have wired high speed Internet. There are various WiFi hotspots located within the hotel. WiFi is not available in the meeting rooms.

## WAYS TO SAVE ON YOUR CONFERENCE REGISTRATION

### Special Early Bird Offer!

Receive \$200 off your registration fee if payment is received on or before August 29, 2008.

### PowerPass Discount

PowerPass holders receive an additional \$100 off their registration fee. Not a PowerPass member? Visit [www.StickyMinds.com/PowerPass](http://www.StickyMinds.com/PowerPass) to learn more.

### Alumni Discount

STAR alumni receive up to an additional \$200 discount off their registration fee.

### Certification Training + Conference

If you attend the Software Testing Certification Training course AND the Conference, you save an additional \$200. See page 6 for course details.

For **Group Discounts** or more details on our discount policy, please contact the Software Quality Engineering Client Support Group at [sqeinfo@sqe.com](mailto:sqeinfo@sqe.com) or 888.268.8770 or 904.278.0524.

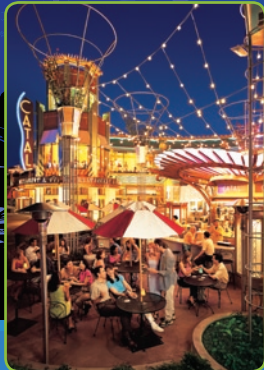


# SOFTWARE TESTING

## ANALYSIS & REVIEW

*The Greatest Software Testing Conference on Earth*

Anaheim,  
**California**



Sept. 29–Oct. 3, 2008  
**Disneyland® Hotel**

As to Disney photos, logos, properties: ©Disney

**GROUP DISCOUNTS  
AVAILABLE**

**Over 98% of  
2007 Attendees  
Recommend  
STARWEST  
to Others in  
the Industry**

**THE TESTING EXPO**  
October 1–2, 2008

*Visit Top Industry Providers Offering  
the Latest in Testing Solutions*

**[www.sqe.com/STARWEST](http://www.sqe.com/STARWEST) REGISTER EARLY AND SAVE \$200!**

**Conference Sponsor:**



**Industry Sponsors:**



**Media Sponsors:**



**CONFERENCES**

330 Corporate Way, Suite 300  
Orange Park, FL 32073