

SEAPINE AGILE EXPEDITION™

explore Agile in an
eBook Adventure

eXplore Agile in an eBook Adventure

The Seapine Agile Expedition is a **free** eBook-based learning series that provides a fun and informative overview of **Agile development**.

New to Agile? Discover how Agile boosts the performance of your entire team—developers, quality assurance, product owners, and ultimately, your customers—through timely communication and collaboration.

Already using Agile? Learn how to meet deadlines, respond to change, reduce risk, and deliver what you promise through Agile development powered by Seapine ALM.



www.seapine.com/betteragile

 Seapine Software™

NOVEMBER 14-19, 2010

ORLANDO, FLORIDA



AGILE DEVELOPMENT PRACTICES CONFERENCE

East

Agile approaches offer a framework of best practices to help you and your company be more effective and respond to changing user and business priorities—all while delivering a higher-quality product. Join us at **Agile Development Practices** to learn how to embrace change, improve your leadership skills, and take back innovative approaches to empower your team and organization to greater results.

Click here to learn more about the conference!

Register using promo code **BSMD** and save up to **\$200**. Use that code by October 15th and double your savings for up to **\$400!**

Conference Sponsor:



www.sqe.com/adpeast

100% OF 2009 ATTENDEES RECOMMEND THE AGILE DEVELOPMENT PRACTICES CONFERENCE

NOVEMBER 14-19, 2010

ORLANDO, FLORIDA



Agile approaches offer a framework of best practices to help you and your company be more effective and respond to changing user and business priorities—all while delivering a higher-quality product. Join us at **Agile Development Practices** to learn how to embrace change, improve your leadership skills, and take back innovative approaches to empower your team and organization to greater results.

Click here to learn more about the conference!

Register using promo code **BSMD** and save up to **\$200**. Use that code by October 15th and double your savings for up to **\$400!**

Conference Sponsor:



www.sqe.com/adpeast

100% OF 2009 ATTENDEES RECOMMEND THE AGILE DEVELOPMENT PRACTICES CONFERENCE

September/October 2010

\$9.95

www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to

StickyMinds.com

THE COST OF FREE
Open source obligations

FIND A FIT
Project management
in Scrum

SIMPLIFY YOUR COMBINATORIAL TESTING



SOFTWARE: WORK FUN?
TAKE OUR SURVEY!

**“We need to make sure that
our company's most important
business gets done every
two weeks.”**

Erik Huddleston

**CTO
Inovis**



“Our business is dependent on tight turnaround times. **Rally makes it easy to juggle scope, hours and resources, so we can share up-to-the-minute project information with all of our stakeholders.** Everyone can see priorities, status and roadblocks to save time and money.”

See how development organizations like Erik's are delivering software the Agile way: www.rallydev.com/videos/

HOW HEAVY IS YOUR CLOUD?

Find out with **QA Wizard Pro**, now with **Load Testing**.

Will your cloud or web application be able to handle the load once it goes live? Will it crash or return errors to your users? Will it slow down for all users, or just bog down during specific functions?

Find out if your web app can handle the real world before you launch with **QA Wizard Pro's** new load testing functionality.

Load testing with Seapine's **QA Wizard Pro** lets you identify where your bottlenecks occur—memory, CPU, network, or database—and what it takes to break your application.

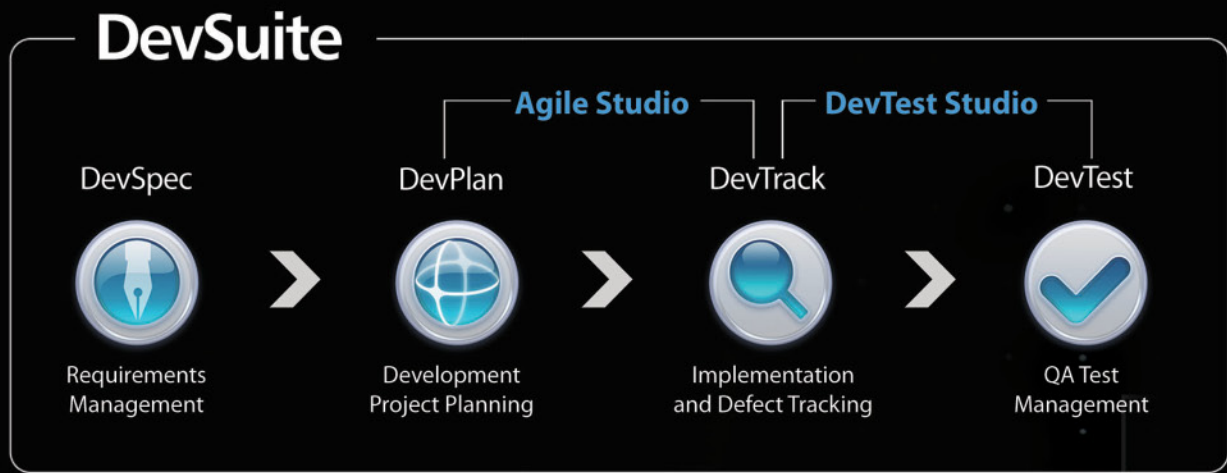
Now one application meets all your automated testing needs—regression testing, functional testing, and load testing.

Download **QA Wizard Pro** at www.seapine.com/betterLT and try it today!



Gain Full Traceability

Requirements Driven Quality Management



A Modular Approach to ALM

- DevSuite can be purchased and deployed as a complete ALM solution or as individual modules
- Convenient bundles provide solutions for quality management (DevTest Studio) and Agile development (Agile Studio)
- Individual modules and bundles can be easily expanded when you need additional functionality

Agile Ready

- DevSuite is a hybrid platform that allows you to mix elements from multiple methods, including traditional development, to achieve the right balance for your team
- Out-of-the box methodology templates for Scrum, XP, Test Driven, Waterfall, and Iterative development



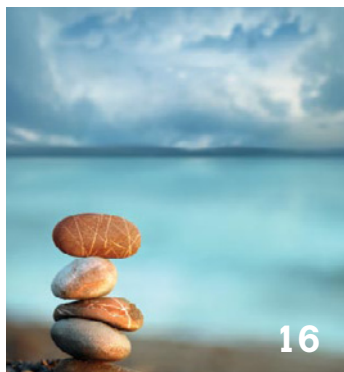
Download your FREE 30-day trial of DevSuite now at: www.techexcel.com/downloads

www.techexcel.com

1-800-439-7782



CONTENTS



16

Take the
Better Software
magazine digital
survey.
This month's topic:
Software for Fun.
pg. 15

in every issue

Mark Your Calendar **4**

Contributors **8**

Editor's Note **9**

Virtual Resource Shelf **14**

Digital Survey Results **15**

Product Announcements **33**

FAQ **34**

Ad Index **36**

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 six issues.

Visit www.BetterSoftware.com
or call 800.450.7854.

features

16 COVER STORY **SIMPLIFY YOUR COMBINATORIAL TESTING**

Combinatorial testing is effective for testing multiple, non-sequential inputs that affect a common output in complex software. But, it's easy to misapply it or become a slave to the output. Learn to overcome limitations and benefit fully from this technique.

by Bj Rollison

22 LICENSE TO OPEN SOURCE

Open source is widespread and growing in many software development organizations. While there's no purchase cost, the code does come with license obligations. Understanding open source from an intellectual property perspective can help avoid downstream legal consequences arising from unmanaged licenses.

by Kamal Hassin and Katherine Chin Quee

26 THE ROLES OF THE PROJECT MANAGEMENT OFFICE IN SCRUM

Successfully adopting Scrum entails understanding and perhaps adjusting the role of the project management office (PMO), whose workers are often resistant to the lighter-weight process. But, they can become a critical part of agile success. Discover how an agile PMO works.

by Mike Cohn

columns

11 TECHNICALLY SPEAKING **COPELAND ON WEINBERG** • *by Lee Copeland*

Lee Copeland and Jerry Weinberg have crossed paths—both on page and in person—many times over the years. Here, Lee reflects on some of those meetings and their valuable lessons.

12 INSIDE ANALYSIS **HEARING "NO"** • *by Rick Brenner*

"No" can be disappointing. Sometimes we have difficulty hearing or dealing with No. Can we learn how to cope with No with less pain and angst? Can we learn how to prevent No at least some of the time? Yes and yes!

35 THE LAST WORD **PRODUCT OWNERS SHOULD CARE ABOUT QUALITY** • *by Roman Pichler*

Product owners often view quality as an ugly duckling—necessary to ship software, but nerdy and a drag. Instead, they should be guardians of quality. Only when quality meets functionality is lasting value created.

MARK YOUR CALENDAR



software tester certification

www.sqetraining.com/certification

September 26–28, 2010
San Diego, CA

October 5–7, 2010
Baltimore, MD

October 12–14, 2010
Raleigh, NC
Omaha, NE

October 18–20, 2010
San Francisco, CA

October 19–21, 2010
Atlanta, GA
Pittsburgh, PA

October 26–28, 2010
Austin, TX
Columbus, OH

training weeks

www.sqetraining.com/public

Testing
October 18–22, 2010
San Francisco, CA

November 8–12, 2010
Tampa, FL

conferences



STARWEST 2010
Software Testing
Analysis & Review
www.sqe.com/starwest
September 26–October 1, 2010
Hilton San Diego Bayfront
San Diego, CA

Agile Development Practices East
www.sqe.com/adpeast
November 14–19, 2010
The Rosen Centre
Orlando, FL

STAREAST 2011
Software Testing
Analysis & Review
www.sqe.com/stareast
May 1–6, 2011
Rosen Shingle Creek
Orlando, FL

Better Software Conference
www.sqe.com/bsc
June 5–10, 2011
Caesars Palace
Las Vegas, NV

Agile Development Practices West
www.sqe.com/adpwest
June 5–10, 2011
Caesars Palace
Las Vegas, NV

BETTER SOFTWARE

Publisher
Wayne Middleton

President
Drew Thoenig

Vice President of Publishing
Holly N. Bourquin

Editor in Chief
Heather Shanholtzer

Editorial

Managing Technical Editor
Lee Copeland

Online Editor
Francesca Matteu

Online Editor
Joseph McAllister

Production Coordinator
Cheryl M. Burke

Design

Creative Director
Catherine J. Clinger

Advertising

Sales Manager
Shae Young

Sales Manager
Sonia Lavin

Production Coordinator
April Evans

Circulation and Marketing

Circulation Coordinator
Jamie Green-Gago

Product Marketing Manager
Diara A. Sullivan

Marketing Coordinator
Stephanie Fender

A PUBLICATION OF
SOFTWARE QUALITY ENGINEERING



CONTACT US

Editors: editors@bettersoftware.com

Subscriber Services:
info@bettersoftware.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

Automate Your UITesting with Ranorex





Object-based Capture & Replay Editor

- ✓ Maintainable recordings via the actions table editor
- ✓ Integration of Ranorex repositories



Automated Testing of Web & Windows Applications

- ✓ Winforms / C# / VB.NET
- ✓ WPF / Silverlight / Win32 / MFC
- ✓ Flash / Flex / Web 2.0 / AJAX /  / 



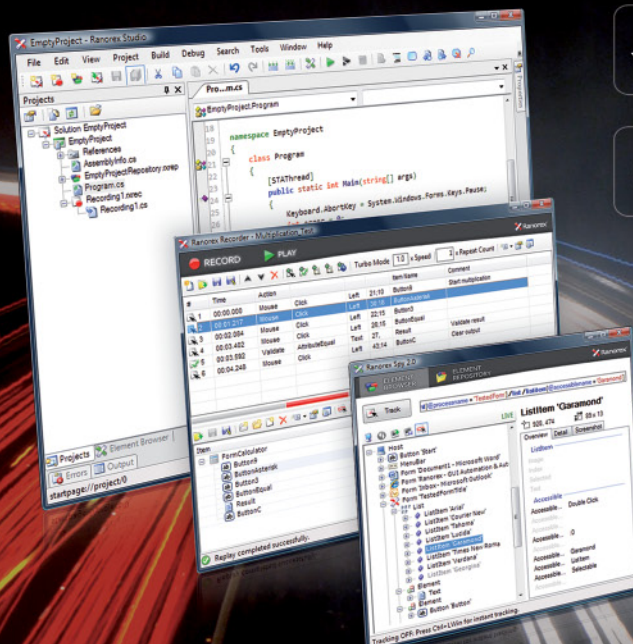
Maintainable UI Object Repositories

- ✓ Easy to maintain all types of UI objects
- ✓ Separate test automation from UI identification



Write tests in C#, VB.NET and IronPython

- ✓ Automatic and flexible code generation in C#, VB.NET and IronPython
- ✓ All-on-one test environment with code editor, code completion and debugging



Get your 30-day Trial
www.ranorex.com



Watch Demo Video
www.ranorex.com/demo

END-TO-END INTEGRATION

RELENTLESS TRACKING

FAST FORWARD

INTELLITRACE™

ACTIONABLE BUGS

PLAN-TEST-TRACK TABS



BEAT THE BUGS. ROCK THE TEST.

**ACCELERATE DEPLOYMENT AND REDUCE COSTS
WHILE INCREASING TEAM CREATIVITY AND PRODUCTIVITY.**

The **Microsoft® Visual Studio® 2010** development system, with **Microsoft Visual Studio Test Professional 2010** and **Microsoft Visual Studio Lab Management 2010**, can help your team improve collaboration, reproduce bugs, automate repetitive tasks, and reduce the costs of setting up and maintaining multi-machine test environments.

Test Professional 2010 enables your teams to:

- **Increase collaboration** by breaking down the silos between test and development.
- **Eliminate waste** across your application lifecycle through simplified test planning, manual test execution, and rich, actionable bugs.
- **Extend capabilities** by combining **Test Professional 2010** with **Lab Management 2010** to extend test capabilities in virtualized lab environments.
- **Enable streamlined testing** with **Lab Management 2010** to rapidly provision virtual lab environments at a known state for test execution and build automation, effectively reducing wasted time and resources in your development and test lifecycle.

"We're seeing a 20 to 30 percent productivity gain with respect to quality assurance-related processes, such as testing and bug fixing."

Jorge Rodriguez Brizuela
Development Supervisor,
IT Finance, Flextronics

Reduce total cost of ownership on virtualized labs, eliminate waste in your application lifecycle, and help accelerate your development and test processes with the combined power of **Test Professional 2010** and **Lab Management 2010**.

EXPLORE AND EXPERIENCE THE POWER



Microsoft®

Visual Studio® Test Professional 2010

Explore the solution at www.microsoft.com/visualstudio/test.

Buy at www.microsoft.com/visualstudio.

Microsoft




RICK BRENNER, principal of Chaco Canyon Consulting, works with people needing state-of-the-art teamwork in problem-solving organizations producing complex products and with organizations that want stronger relationships among their people. His interests are personal and organizational effectiveness in abnormal situations, such as dramatic change, enterprise emergencies, and high-pressure projects. His articles and weekly newsletter are available at www.chacocanyon.com.




MIKE COHN is the founder of Mountain Goat Software, where he teaches and coaches on Scrum and agile development. He is the author of *Succeeding with Agile: Software Development with Scrum*, *Agile Estimating and Planning*, and *User Stories Applied for Agile Software Development*. Mike is a founding member of the Scrum Alliance and the Agile Alliance. He can be reached at www.mountaingoatsoftware.com.



LEE COPELAND has more than thirty years of experience in the field of software development and testing. He has worked as a programmer, development director, process improvement leader, and consultant. Based on his experience, Lee has developed and taught a number of training courses focusing on software testing and development issues. He is the managing technical editor for *Better Software* magazine, a regular columnist for StickyMinds.com, and the author of *A Practitioner's Guide to Software Test Design*. Contact Lee at lcopeland@sqe.com. 



RICK CRAIG has a wide range of experiences as a tester and test manager. Currently a consultant with SQE Training, he specializes in metrics, management, and process improvement. Rick has spoken at conferences around the world every year since 1984, including the Better Software conferences and every STAR conference. He is the co-author of *Systematic Software Testing*, a regular columnist for *Better Software* magazine and StickyMinds.com—and a Colonel in the USMC Reserve. 



Director of R&D and product management at Protecode (protecode.com), **KAMAL HASSIN** is a thought leader in the area of open source licensing and the author or co-author of a number of papers on software intellectual property management. Kamal has a bachelor of engineering degree and a master in technology innovation management from Carleton University. He can be reached at khassin@protecode.com.



A leading Scrum and agile product management expert, **ROMAN PICHLER** has a long track record of teaching and coaching product owners. For nearly a decade, he has helped companies transition to agile. Roman is the author of *Agile Product Management with Scrum* and a frequent speaker at international conferences. Find out more at www.romanpichler.com.



KATHERINE CHIN QUEE is a legal consultant at Protecode, a provider of software intellectual property audit services and solutions for software governance and intellectual property management. She has a bachelor of commerce degree with a concentration in information technology management, a juris doctor from the University of Toronto, and a master of laws in intellectual property law from Benjamin N. Cardozo School of Law.



BJ ROLLISON is a principal test architect with Microsoft's Engineering Excellence team, which is responsible for the technical training of Microsoft's engineering community. He teaches software testing and test automation courses at the University of Washington, is co-author of *How We Test Software at Microsoft*, and enjoys developing test data generation tools, which are available at www.testingmentor.com. You can reach Bj at bj.rollison@testingmentor.com.



ALL YOUR FUTURE LIES BENEATH YOUR HAT

Hi, everyone. Don't mind the unusual mug in the corner over there.

Heather Shanholtzer is away for a couple of months, enjoying her new role as mother to a beautiful baby daughter. She will return to these pages in short order, and all of us at *Better Software* magazine wish her very hearty congratulations and look forward to having her back.

Since she's been away, her coworkers have taken to sharing her many metaphorical hats. When it comes to roles around the office, an editor in chief is a virtual millinery shop.

But, while we deal with having more hats than heads, some offices are concerned with the opposite problem. In this month's "The Roles of the Project Management Office in Scrum," Mike Cohn takes a look at a common issue with many agile adoptions: If everyone's sharing project management responsibilities, where do the project managers fit in? As it turns out, they've got all sorts of hats to fill—though those hats may look a little different from the ones they're used to wearing.

In "Simplify Your Combinatorial Testing," Bj Rollison takes a powerful technique for testing the effects of many inputs on a common output and breaks it down into a simpler, more manageable approach. And, in "License to Open Source," Katherine Chin Quee and Kamal Hassin review some of the finer points of open source software licenses. (If your definition of "open source" is "free," don't miss this article.)

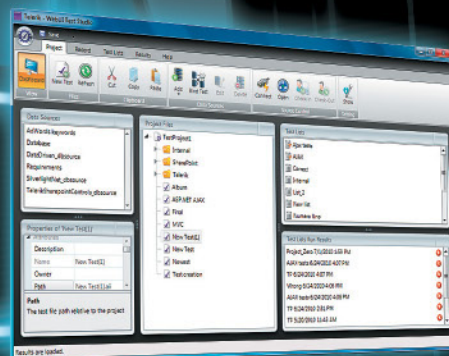
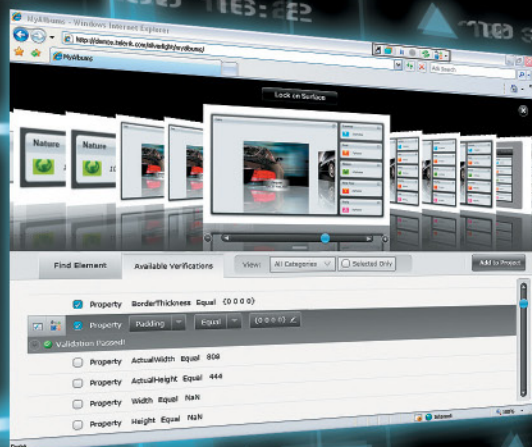
As always, no matter what hat you wear to read it, we hope you enjoy this issue of *Better Software* magazine. Email me to let me know how you've put this issue to work for you.

Happy Reading!

Joey McAllister

jmcallister@sqe.com

Testing is Now Radically Easier



Telerik WebUI Test Studio

Easy Test Creation

- Test any web app – HTML/AJAX/Silverlight
- Intuitive point-and-click recording
- Record once, run against multiple browsers

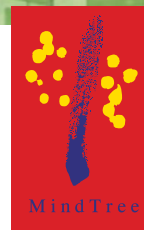
Easy Test Maintenance and Management

- Element abstraction and reuse
- Scheduling, execution and results reporting
- Seamless QA – Developer collaboration

Download your free 30-day trial at www.telerik.com/RadicallyEasier

 **telerik**
deliver more than expected

predictable quality through independent testing



Quality lies in the eyes of the beholder. According to Jerry Weinberg - "Quality is value to some person". As per J.M. Juran - "Quality is fitness for use". This article is not about our views on quality. Whatever criteria we set for determining quality, we have to ensure that we achieve it time and again - this is what we believe predictable quality is. This article is about what it takes, from an independent testing perspective, to achieve predictable quality each time, every time.

Testing is a multi-faceted activity where many entities have to operate in synchronization. On one hand, we need to have a high caliber testing team with the requisite technical and domain expertise. On the other hand, it is of utmost importance that all the activities are governed by a set of mature processes, metrics and audit mechanisms to ensure the repeatability of accomplishment which is essential for achieving predictable quality. This is the "Testing Methodology" that has to be put to practice which we will focus in this article.

Needless to say, challenges have to be overcome to have a large team comply with a standardized set of processes and follow a common set of guidelines, templates and checklists. At the same time, we have to be careful not to cross the thin line between process compliance as an enabler for predictable quality versus being an overhead. Let's take a case study to exemplify the sayings of Hyman Rickover "Good ideas are not adopted automatically. They must be driven into practice with courageous patience."

As a part of testing center of excellence (TCoE) implementation for one of our customers who is a leader in the asset management domain, we evaluated the maturity of the current testing processes in the pre-transition phase. The portfolio consisted of 70 applications for which an integrated testing solution had to be implemented. The existing team had 110 testers distributed into multiple product lines. We spent two weeks in discussions with close to 60 stakeholders across various divisions and roles.

There were multiple gaps that were uncovered; the key challenge was in terms of inconsistencies in certification of application quality across releases. Documented test process definitions and governance mechanisms were in place and were followed, but their efficiency and effectiveness was questionable. We will elaborate on a few important areas

- **Metrics** - They were development centric and more applicable at project level, and validity of data was a concern. There weren't any baselines; hence there was lack of clarity in inferring the data and corrective actions to be taken. Thus, lack of motivation and feeling of overhead in terms of metrics data collection was understandable.
- **Audits** - They were more oriented towards process compliance checks, rather than evaluation of health of testing.
- **Support system** - There were no common set of tools for test case management, test result tracking, metrics management and knowledge collaboration. Each project used its own set of tools; mostly excel spreadsheets, which had its own collaboration challenges. Thus, there were limitations in reusability of test artifacts and centralized governance leading to inconsistencies in certification of application quality.

Next steps involved plugging of identified gaps in a phased manner over the transition and steady state phases. We will limit ourselves in describing the approach of evangelizing MindTest™, our way of testing, with respect to the above mentioned areas.

MindTree's test methodology is driven by organizationally aligned objectives on "Customer Satisfaction" and "Project Performance". Over the years, we learned the hard way that to achieve excellence, the processes have to be "practitioner defined and practitioner refined", and the mindset has to be changed from "destruction oriented" to "prevention oriented".

Measurement Model

We have seen that many projects are good in collecting metrics; however, they hardly use these metrics for pattern analysis and taking corrective and preventive measures. Also, many successful projects used subjective measurement techniques additionally for qualitative assessment of user opinion and preferences. Based on these experiences and observations, we implemented MindTree's Test Metrics Analysis and Decision (TMAD) and Test Quality Index (TQI) models for objective and subjective measures respectively.

The following snapshot gives an idea of the TMAD model; it has a set of 19 categorized metrics with additional information on computation details, suggested collection frequency, suggestion on related corrective & preventive actions and more importantly, the relationships between the metrics.

Metrics	Goal	Associated Queries	Be aware of
Defect Containment	1. Improve Testing Quality	1. Are there clear entry and exit criteria between phases?	1. Blame game
Effectiveness	2. Prevent defect leakage	2. Was the scope of testing for each phase clear?	2. Manipulation of defect classification criteria

The Test Performance Index (TPI) was calculated based on the weighted average score of the metrics.

The TQI is a proprietary tool for self assessment & scoring, emphasizing on key quality expectations of customers. The focus is on how well a task has been executed, rather than the evidence of whether the task has been executed or not. It is for test quality improvement and the intention is not to project high scores to external stakeholders. The following snapshot gives an insight.

No	Index Parameter	Resp	Weightage	Rating	Comments
7	Quality of Test Design and Quality of Test Cases			1	2.15
7	Does the test design map to the test objectives from Customer?	Y	2.00		
8	Is the test case prioritization done for current release?	N	1.00		Prioritization is done but not updated for current release
9	Are the test cases identified/classified for Automation?	NA	1.00		Identification of Automation Test Cases is not in scope
10	Have you written test cases for post production bugs if any?	N	1.00		Few have been linked not all bugs linked to test cases
11	Are the test design and test cases approved by the Customer?	N	2.00		There is delay in the loop to get the test cases reviewed by customer
12	Have you applied the Test Case and Design Checklist?	Y	1.00		
	Please rate your overall performance of project execution in this area, on a scale of 1 to 5 (1 being the lowest, and 5, the highest)		3.0	2.58	Some areas we have marked as N are partially done so the subjectivescore is given as 3.0

Month Name	Cur Month	
Test Quality Attribute	Wgt	Score
Quality of Test Strategy & Planning	1	1.63
Quality of Test Design & Quality of Test Cases	2	2.58
Traceability, Quality of Test and Code Coverage	1	1.63
Quality of Test Execution and Result Reporting	1	1.54
Test Metrics, Analysis and Improvements	2	2.44
Tools and Frameworks Usage	2	2.58
Net Rating Total		2.06

Support System

We adopted a set of web based tools for consolidated governance and collaboration that are used in other projects at MindTree. We will describe a few key ones

- **ProcessNet** - Intranet portal that hosts the processes, templates, checklists, guidelines and best practices to execute projects along with workflows to capture and track the process improvement suggestions, customer complaints, escalations and deviations.
- **Test Dashboard** - Central repository for metrics data, captured at a project level which can be rolled up at the TCoe, industry segment and organization level thus providing the flexibility to create multi level benchmarks. This benchmarking becomes useful for future projects in the areas of estimation, productivity and quality predictability.
- **TestLink and RADAR** - Organization wide deployed tools for test and defect management respectively.
- **ProjectSpace** - Centralized web based tool serving as a persistent base for project specific knowledge and release management fostering collaboration and effective delivery in a multi location environment.

Review Mechanisms

Apart from the process compliance audits and project management reviews that were already in place, we introduced “Test Audits”. In these audits, we evaluated the testing related metrics like defect containment and reporting effectiveness, test coverage and effort/schedule variances, savings in test effort/cycle through process optimization & automation, issues and risks.

One of the immediate perceived benefits post adoption of MindTest™ was availability of right information at the right time to the right people. The following impacts were observed during the steady state phase

- **Test dashboard** enabled benchmarking productivity for estimation that helped in reducing the UAT defects through better effort allocation for testing. It provided a bird's eye view on the overall progress of testing backed by data to senior management which aided in early escalation of risks and issues through quantifiable means and receiving the required support.
- **The TMAD model** helped in justifying the test certificates by quantitative data, take corrective and preventive actions, establish and confirm to quantifiable entry & exit criteria and go-no-go decisions. The TQI helped in continuous improvement of the quality of testing and we were able to foresee TPI based on a TQI prediction model (TQIp).
- **TestLink and RADAR** helped in establishing end to end traceability for test coverage analysis

We observed that after a period of 9 months, the defect containment effectiveness was consistently above 95% and invalid defects below 6%. The ability to justify the go-no go decision via coverage and defect detection rate metrics brought about a culture of excellence and the confidence to achieve predictable quality, each time, every time.

In this article, we have illustrated the process aspects of implementation of a testing methodology. We know how challenging building a high caliber testing team can be, which also plays an important role towards achieving predictable quality. It helps to have pillars of excellence to support the process evangelism and delivery teams, like an “Academy” for competence development or a “Testing Labs”

Copeland on Weinberg

In work and in life, Lee Copeland has found value in the wisdom of friends and mentors like Jerry Weinberg.

by Lee Copeland | lcopeland@sqa.com

I first “met” Jerry Weinberg in 1971 through his book *The Psychology of Computer Programming*. [1] There had never been a book like it. In the preface, Jerry wrote, “This book has only one major purpose—to trigger the beginning of a new field of study of computer programming as a human activity, or, in short, the psychology of computer programming.”

Until that time, the focus of programming was on hardware and software, not the people involved in doing the work. Jerry called on us to “look upon the programmer as a human being, rather than just another of the machines.”

Before writing this column, I reread *Psychology* and was amazed to find it was the origin of many ideas I’ve found useful over the years—the value of reading programs in addition to writing them; that where there are multiple users, there are always multiple requirements; Fisher’s Fundamental Theorem; that most of what we know about psychology is based on the study of college freshmen taking psychology courses; that programming is a social activity, not a solitary one; that one specific personality test often used to select programmers was actually developed to diagnosis mental illness; and that both complete requirements and complete testing are impossible.

I first met Jerry and his wife Dani in person at their Problem Solving Leadership (PSL) workshop. PSL is an intense, weeklong combination of learning, experimentation, reflection, feedback, and, for me, personal discovery. My favorite recollection, other than Jerry’s enjoyable stories, was my choice to be an observer rather than a participant in the VerseWorks exercise. I learned to focus on what others were doing—their words, expressions, and interactions—in a way I had never done before. Learning to become an observer has helped me immensely over the years. In addition, an introduction to Myers-Briggs personality types helped me understand that all those weird people in the world who don’t think and act like I do are really OK. They just see the world differently.

In 2008, seventeen of Jerry’s friends each contributed a chapter to *The Gift of Time: Celebrating the Work of Gerald M. Weinberg*, edited by Fiona Charles. Each author had been touched by Jerry’s kindness and the gift of his wisdom and time, and each wanted to recognize and repay that gift.

Many years ago, I had a difficult problem at the office.

I was at a standstill. I called Jerry at his home in Colorado, hoping for perhaps twenty minutes of advice over the phone. After explaining my situation, Jerry thought for a while and replied that my problem was too complicated for a short phone call. He invited me to come to his home in Crested Butte for a day or two to fully discuss it. I explained that I had no money in the budget to pay his consulting fee. Jerry laughed and said, “It’s not consulting. It’s two friends solving a problem.”

In November 2009, Jerry posted this on his Web site:

**“I learned to focus on
what others were doing—
their words, expressions,
and interactions—in a way
I had never done before.”**

I have a thymic carcinoma, a rare and aggressive cancer that has perhaps a 25% one-year survival rate if treated aggressively. Apparently this nasty thing can move very quickly throughout my body. The MDs have just given me details on the upcoming treatment plan: It’s not treatable, at least not curable. Surgery is not possible, and chemotherapy might at best alleviate my suffering and prolong my life. I’m going to cut short almost all of

my communication and say a tentative good bye to all of my loved ones, unless a miracle occurs. [2]

Jerry has completed therapy and just a short time ago wrote, “More good weeks! I’ve now been feeling well enough and sharp enough to get back to the work on my novels. All in all, good signs of my recovery.”

Jerry concludes *Psychology of Computer Programming* with these words: “Is what we are doing with computers worth doing? Is what you are doing with computers worth doing? Because computers are such fascinating beasts, because programming is such a game, such a joy, we who program computers are in danger of becoming the unwitting pawns of those who would use our toys for not-so-playful ends.”

We could ask another question: Is what we are doing with our lives worth doing? Is what you are doing with your life worth doing? All of Jerry’s friends are grateful for what he has done with his life—his caring, kindness, wisdom, and love. Jerry, we at *Better Software* magazine and our readers all wish you luck. {end}

**Sticky
Notes**

For more on the following topic go to
www.StickyMinds.com/bettersoftware.

■ References

Hearing “No”

Don't let “No” put the brakes on an otherwise smooth project. Learn how to prevent it or, if it's inevitable, how to cope with it.

by Rick Brenner | rbrenner@chacocanyon.com

You've been working with a group of stakeholders to forge consensus on a project issue. Some want exactly what others don't want, some refuse to reveal their private agendas, and some seem to change their goals almost at random. At times, you've felt that the group was close to agreement, only to be disrupted when someone on high changed the external constraints.

It's been a little frustrating.

Work life sometimes delivers disappointments, often in the form of No. Some of us have difficulty hearing No or dealing with it once we do hear it. And, sometimes, No arrives so frequently that we exhaust our ability to cope with it.

In the tips that follow, I'll use the term No-giver to refer to the person or people who said No and *recipient* to refer to the person or people receiving the No.

To understand the full range of choices recipients have, it's useful to analyze the situation from three perspectives: within, between, and among.

Within describes the recipient's internal response. Some of us conceal our internal responses from others and even from ourselves.

Between pertains to relationships. There are tight connections between the method of delivery, how it's received, and past experiences shared by the No-giver and recipient.

Among includes connections in the larger context—the organization, its people, and their perceptions. When we consider stakeholders as a whole, we're usually considering the larger context.

Suppose that you're presenting to a general manager's staff a plan proposed by IT. The plan doesn't provide everything the general manager wanted by the desired dates, but it does deliver everything eventually. IT has explained why this is necessary, and it's your task to present their case as a starting point for further negotiation. After you've clarified general management's objections, you go back to IT to help them understand why adjustments to their proposal are needed. At times, you feel a lot like a high-level diplomat shuttling back and forth between the leaders of two warring countries who speak entirely different languages. (Throughout the article, you'll find references to examples that are available in the StickyNotes.)

Within: The Self

No can arrive with a thud: “No can do” or “When pigs fly.” Or, it can land gently: “Not at this time” or “We've decided to go in a different direction.”

No can rock our world, or it can be a useful prod to make alternative plans. How we respond can make the difference between success and decades of pointless wandering.

Here are four techniques for enhancing your inner response to No.

Remove time pressure: While formulating a response, time pressure limits clear thinking. Even if an immediate Yes is needed, taking time to think does little harm. We usually do much better if we know in advance how much time we can spare before we must act.

If we enter the conversation knowing that we do have a little time to think, we're more likely to respond creatively to No. (See Example 1)

Question your own analysis of the consequences: Failing to consider the consequences of No carefully enough makes receiving No difficult, and we're more likely to respond to No emotionally or to reject it. Usually, that leads to trouble.

The pathway opened by No can sometimes be more desirable than the pathway opened by Yes. Examine it courageously. (Example 2)

Distinguish your request from your Self: Sometimes we experience No as the answer to the question “Am I a good person?” Rarely is this the issue. Usually, No is related more directly to what was requested.

Distinguishing your request from your Self provides protection from imagined attacks or criticism and helps you recognize them when they're real. (Example 3)

Be prepared for No: When No comes as a total shock, hearing it is truly challenging. To avoid the shock, prepare by asking four simple questions: What if the answer is No? How long will it take us to find another approach? Is another approach actually impossible, or is the search for it simply distasteful? Is the organization harmed by No, or is it mostly my personal problem?

Answering such questions in advance helps you control emotional responses. (Example 4)

“‘No’ can rock our world,
or it can be a useful prod
to make alternative plans.”

Between: Relationships

Relationships matter. If the recipient has long been a target of abuse by the No-giver, fear might be a likely response. If the No-giver and recipient have a history of constructive collaboration, honest discussion is likely to ensue. Here are five practices that help build or maintain constructive relationships with No-givers.

Seek to understand why: Our first response to No might be emotional: hurt, consternation, or anger. To recover a more serene state and move toward alternatives, seek to understand the No-giver's view. Even though the No-giver might be unable or unwilling to be candid, our response to No is more effective if we understand the reasons for the No. (Example 5)

Take the broad view: To understand what led to No, apply the Johari Window. [1] The Open factors alone might not explain it, because some factors might be Hidden from the recipient, the No-giver might be Blind to other factors, or critical information might be Unknown to both. The Blind and Unknown quadrants of the window are most likely to produce convergence.

In responding, avoid the fundamental attribution error. Don't attribute the No to the No-giver's character defects without solid evidence. (Example 6)

Never threaten: One popular, threatening response to a subordinate's No is "If you can't do it, we'll find someone who can." A threatening response to a superior's No is "Just remember I warned you." Threats rarely produce conversions. At most, they produce compliance, and they almost always damage relationships.

Anger, fear, and feelings of powerlessness can lead to threats. When you feel the urge, clear thinking may be difficult. Recover control or suspend the conversation. (Example 7)

Prevent, rather than respond: Hearing No when it arrives is useful, but preventing it is even better. Here are four effective strategies:

- Yes is more likely if the relationship is strong. Keep it strong.
- Ask only for what's likely to be granted.
- Prepare the decision-maker by providing whatever is needed for Yes.
- Create in the decision-maker an aversion to the consequences of No.

By working toward Yes in advance, you'll learn more about why Yes might be appealing to the decider, and that can help you craft your request. (Example 8)

Ask whether the No-giver is empowered to say Yes: If we know the No-giver lacks the authority to say Yes, we can prepare our response to the No. If the No-giver has no choice, investing in prevention might be pointless. In determining whether No-givers have real choices, remember that their constraints might include factors external to the organization. (Example 9)

Among: The Larger Context

The larger context includes everything beyond yourself and your relationships: laws, regulations, procedures, and organizational politics.

Know whether there is an appeal path: Knowing that recourse is possible can help you remain centered when No arrives. Familiarity with the appeal path can help you frame the original request.

No-givers dislike being overruled. Construct your request so as to enhance the likelihood of a successful appeal. (Example 10)

Know whose No it is: Sometimes the No is directed at denying something to someone else or to another effort. In such cases, responses that leave the true target untouched may be ineffective. Seek alliance with the true target, or adjust your request to minimize entanglement with the target. (Example 11)

Understand reality: Sometimes the No is determined by the laws of nature, legal requirements, or financial constraints. Financial constraints are the least objective, but there are limits even there. A realistic perspective can help you to

continued on Page 14

THE Secret TO Designing Products Customers LOVE



Read the findings
from Aberdeen at
jamasoftware.com/go



build great products™

Virtual Resource Shelf

Author recommended books, blogs, gadgets, Web sites, and other tools for building better software

Q: What is one of your favorite books about planning or problem solving?

Software Tools

by Rian Kernighan and P.J. Plauger

It shows how two master programmers think through solving a problem.

—Mike Cohn

A Guide for Lawyers and Policymakers

by Paul Brest and Linda Hamilton Krieger

—Katherine Chin Quee

Facilitator's Guide to

Participatory Decision-Making

by Sam Kaner, et al.

—Roman Pichler

Rapid Development: Taming Wild Software

Schedules

by Steve McConnell

So much has been written about software but rarely do you see the clarity and relevance that Steve McConnell brings to the topics in his books.

—Kamal Hassin

The Last Place on Earth: Scott and Amundsen's Race to the South Pole

by Roland Huntford

It's a gripping tale of leadership, teamwork, conflict, risk management, organizational politics, scope creep, project failure, burnout, resource management, and the comparative advantages and limitations of agile processes and conventional processes.

—Rick Brenner

MEMORY LEAK FOUND



MemoryScape enables rapid, visual analysis of memory leaks, overruns and usage

● C, C++, Fortran
● Linux, Unix, Mac

Download trial version:

<http://www.totalviewtech.com/download>

TOTALVIEW
TECHNOLOGIES
A Rogue Wave Software Company

email: sales@totalviewtech.com

Inside Analysis continued from Page 13

avoid asking for the impossible and to accept No when Yes is impossible. (Example 12)

Does No really break my world?: No might be the end of any plans that assumed Yes, but it usually isn't the end of the world as we know it. To examine the No version of the world objectively, ask, "If I could still accomplish something I wanted, how would I do it now?" The essential question is "How can you re-point yourself toward something else you want?" (Example 13)

Final Words

Because reflection can facilitate learning, conduct a No retrospective after any especially difficult—or especially successful—incident. Reflect on what worked and what didn't—within, between, and among—when you received a No. If you hesitate to reflect on this because such reflection might be a bit painful, that's just your Self telling you No. But, you know how to deal with that, right? **{end}**

Visit **StickyMinds.com**
to comment on this article

Sticky Notes

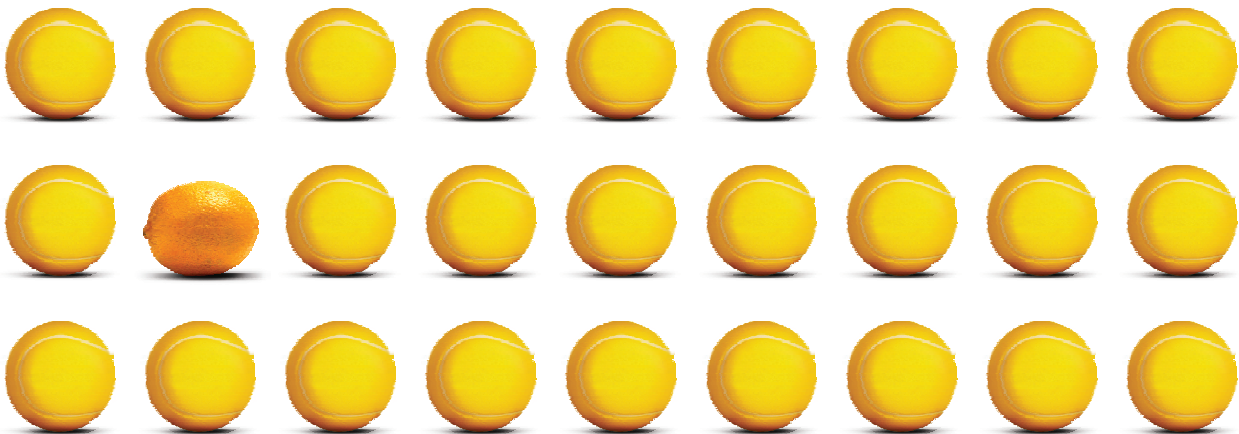
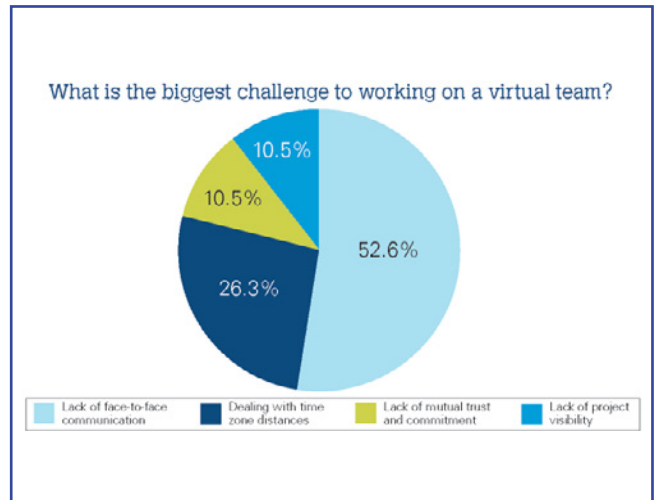
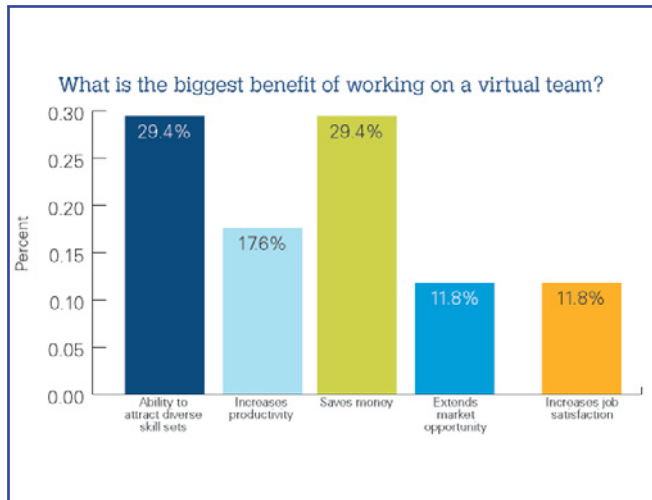
For more on the following topics go to www.StickyMinds.com/bettersoftware.

- References
- Examples

Participate in our new survey: Do you work on software apart from your day job?
www.stickyminds.com/softwarefun

Last Issue's Survey Results:

Q: Do You Work on a Virtual Team?

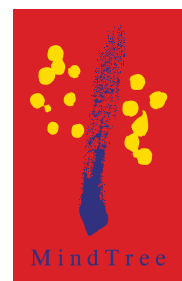


IF ONLY QUALITY ASSURANCE WAS THIS EASY

At MindTree, we have a keen eye for quality, and, an ROI culture that understands that time is money. MindTree can help you minimize risk, costs and time to market, by making quality as predictable as your key business processes. We call it **Predictable Quality through Independent Testing**. Our clients call it sound business sense. So get in touch, and put our talent to the test.

www.mindtree.com

Meet us at Booth 4 & 5 at STARWEST 2010





SIMPLIFY YOUR COMBINATORIAL TESTING

by Bj Rollison

ISTOCKPHOTO

You're assigned to test a new text editor program. One feature you need to test is the font dialog illustrated in figure 1, which allows the user to select from among four different fonts, two basic font styles, two effects, six colors, and a range of font sizes as integer values from 1 to 1638. A change in any single variable or any combination of the variables for these seven parameters affects a common output condition—the attributes of the character glyphs in our text editor's rich edit control. This is a typical combinatorial-testing problem, but even this relatively simple feature has approximately 628,608 [1] possible unique combinations (there are actually slightly fewer due some restricted variable combinations as we will later discover). If we wanted to test every possible combination of variable settings, and if each test took approximately one minute to complete, it would require more than 425 days for exhaustive testing of this single feature.

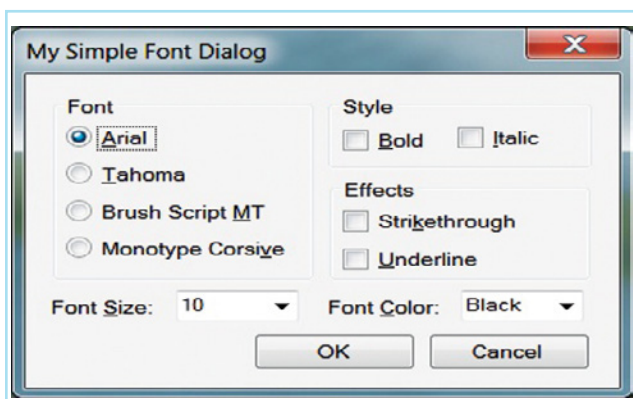


Figure 1: Simplified font dialog feature

Testing combinations of input variables that affect a common output is important because several industry and academic studies [2] conclude that variable interactions are often problematic and a common cause of multi-modal bugs in software. But, it is usually not feasible to test exhaustively all possible combinations of input variables. So, which of the more than 600,000 combinations do we choose to test?

We would likely pick the input variable settings commonly used by our customers, combinations based on historical failure indicators, or those that have exposed problems in the past, and maybe we can intuit a few more interesting combinations. Random selection and exploratory approaches do not provide an easy way of assessing the effectiveness of our test coverage and usually lead to redundant tests. In this situation, we need some way to help us systematically produce a subset of all possible tests that effectively tests important combinations; has a high probability of exposing functional, multi-modal defects and risks; and provides confidence in the test coverage.

Combinatorial Analysis

Combinatorial analysis is an effective functional test technique that can help us systematically evaluate complex relationships of interdependent input variables that directly or indirectly affect some predetermined output state or condition. In order to apply this technique effectively to the font dialog

feature, we must understand how to decompose the feature under test. Feature decomposition for combinatorial testing generally involves identifying the output conditions we wish to evaluate, identifying the parameters that directly or indirectly affect that output condition, and understanding how the variables for each interdependent parameter affect the output condition.

All the parameters on our font dialog affect a common, unique output condition—the properties of the glyphs in the rich edit control of our text editor program. Changing the valid input variable state or value for any of the seven parameters on the dialog will accordingly change the properties of the characters or glyphs in our rich edit control. It doesn't matter if we change one or more variables, or in what order they are changed, because the changes to the glyphs only occur when the OK button is clicked and the appropriate values for the font properties are applied to the applicable text in the rich edit control of our text editor program.

Once we identify a common output condition to test, we need to identify the various input parameters that directly affect that output condition. A little exploration of our font dialog reveals we have a font parameter with four variables (bold, italic, underline, and strikethrough), six basic colors, and a range of font sizes from 1 through 1638.

After identifying the interdependent parameters, we need to analyze how the input variables affect the output condition so we can define appropriate input variables for use in our tests. We should also separate the invalid and valid input variables for negative and positive test cases. In this example, we are only defining the input variables for positive testing of the font feature—i.e., any combination of input variables should appropriately affect the output condition and not cause any adverse behavior.

Since the number of variables for the font, style, effects, and color parameters is relatively small, we should use each unique variable in our tests. But, the font size parameter has a fairly large range of possible valid integer values. Do we need to test all 1,638 values in combination with all the other input variables, and, if not, which values do we choose?

It is probably not prudent to test all 1,638 values in some n-wise order of combinations with all other variables. We could assert that the entire range of valid values is one equivalence partition and any of the 1,638 values within the range will produce the same outcome. Using this approach to design our tests, we might arbitrarily select a limited sample set of values and hard code those specific values for use in our tests.

However, hard coding a small subset of values from a relatively large population of possible values is generally not the most effective approach using equivalence-class test design concepts. One problem with hard coding a variable is that the hard-coded values become static, and static test data loses its effectiveness in subsequent tests using the same test data. Also, hard coding specific values in a large range of values means that we have 0 percent probability of including any other values that are not specified in that range.

Another problem with hard-coded values stems from the criteria used to select from a set of possible values. Typically,

we select values from a set based on historical failure indicators, customer data, and our own biased judgment or intuition of “interesting” values. Of course, any value we identify as important should be included in our tests, but to help reduce uncertainty we should also use scientific sampling methods to select unbiased values from the entire equivalence-class population.

Instead of including the entire range of 1 through 1638 into one equivalence set and selecting a few representative samples, I recommend an alternative test-design approach. First, divide the entire range of font sizes into sub-ranges, or logical subsets that are representative of the entire set. The advantage of dividing the entire range into subsets is that it gives us better distribution of values across the range of possible values. Next, each time an abstract sub-range is specified, randomly generate a value within that defined sub-range for use in that test. Random generation of values within a smaller range increases the variability of values being tested each time the sub-range is specified in a combinatorial test and increases test coverage using a greater number of unbiased samples from the population of all possible values across the entire font range. For example, I defined the following set of valid abstract ranges for the font size input variables:

- *small* indicates a value between 1 and 9
- *nominal* is a value between 10 and 14
- *large* is a value between 15 and 24
- *xLarge* is a value between 25 and 48
- *xxLarge* is a value between 49 and 72
- *xxxLarge* is a value between 73 and 255
- *ridiculouslyLarge* is a value between 256 and 1638.

Now, when a test specifies the *small* font subset, we can randomly select a value between 1 and 9 for that particular test. We (or our automated test) can randomly select a different value for subsequent tests that specifies a small font variable. The logical equivalent subsets you define may be different depending on your perspective, level of confidence, and knowledge of the underlying system. Fewer subsets mean fewer random values are covered in a fewer number of tests. Too many subsets of the entire population increases the number of tests but may decrease the potential for new or interesting information from redundant tests. There is no magic formula, but understanding the variables, how they are used internally, and how they affect the output condition under a variety of situations can help us determine how to define our subsets.

Let a Tool Do the Heavy Lifting

Trying to handcraft a set of pairwise tests in which every variable combination for each pair of input parameters is tested at least once is usually impractical. Fortunately, there are several tools available to help us produce a set of pairwise (or higher order) tests more efficiently. The tools simply read the data we provide and produce a subset of all possible tests

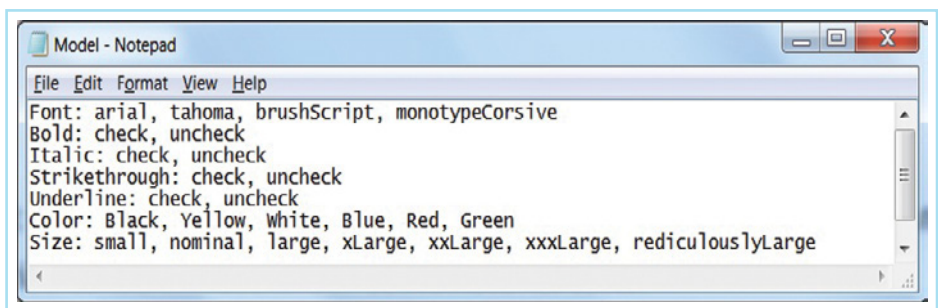


Figure 2: Initial input file of the simplified font dialog for the PICT tool

based on a selected n-wise order of combinations and the variables we define for the feature we are testing. The input to the tool is essentially a model of the parameters, and the variable values that we assert are directly or indirectly associated with the output condition we are evaluating. A comprehensive list of other commercial and free tools is posted on the Pairwise Testing Web site. [3]

Many of these combinatorial testing tools use algorithms based on orthogonal or other types of coverage arrays to produce a set of pairwise (or n-wise) tests. See the StickyNotes for more information on orthogonal or coverage arrays.

For this article, I use the Pairwise Independent Combinatorial Testing (PICT) tool, which is freely available to the public on the Pairwise Web site. The PICT tool requires a text file input that lists the input variables for each interdependent parameter we want to include in our test matrix. Figure 2 illustrates the initial input file that models the parameters and variables we will use in our combinatorial tests.

But, when we analyzed the font feature, we discovered that if the Monotype Corsive font were selected, an event handler in the software set the checked properties either to true for both the bold and italic style checkboxes if either checkbox control were checked by the user, or to false (unchecked) if the user unchecked either checkbox control. Also, Brush Script MT could only have style properties of either bold or bold and italic.

Remember, George Box said, “All models are wrong, but some models are useful.” This initial model of variable inputs is wrong! Using the input file in figure 2 would produce tests with at least one combination of a Monotype Corsive font with bold checked and italic unchecked and Brush Script MT with bold and italic styles unchecked. The tests with these variable combinations would result in false positives because these settings are invalid. This is a great example of why it is of paramount importance always to inspect and validate our models and the output of any testing tool.

Of course, simply modifying the tool’s output is reckless and may inadvertently impact other variable combinations generated by the tool, potentially invalidating or diminishing the effectiveness of the set of combinatorial tests. To solve this problem, we need to be able to customize the tool’s output for invariant or conditional constraints between input variables. The PICT tool includes a BASIC-like programming syntax, allowing us to customize the input data used by the tool. Figure 3 illustrates the three rules we must add to our input model to constrain invalid combinations and give us better control over

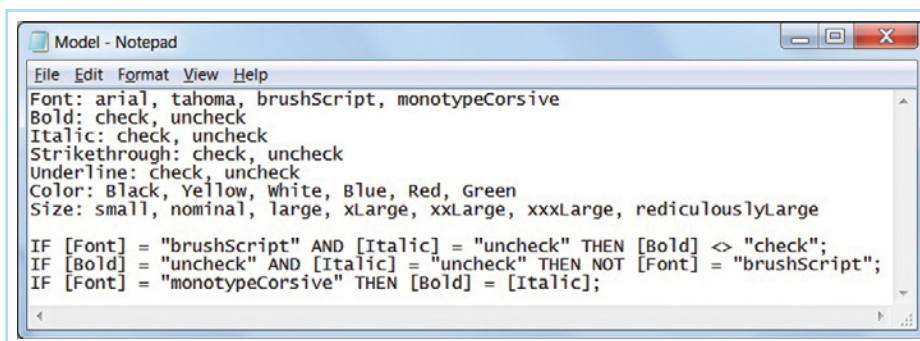


Figure 3: Customizing the model used by the tool to generate the set of combinatorial tests

	A	B	C	D	E	F	G	H
1	Font	Bold	Italic	Strikethrough	Underline	Color	Size	
2	arial	check	unchecked	check	check	White	ridiculouslyLarge	
3	monotype	unchecked	unchecked	check	unchecked	Black	xLarge	
4	brushScript	unchecked	check	check	unchecked	Yellow	large	
5	tahoma	check	unchecked	check	unchecked	White	nominal	
6	tahoma	check	check	check	check	White	xLarge	
7	arial	unchecked	check	check	check	Green	large	
8	brushScript	unchecked	check	unchecked	check	Red	nominal	
9	tahoma	check	unchecked	check	check	White	large	
10	arial	check	unchecked	check	unchecked	Red	ridiculouslyLarge	
11	monotype	check	check	check	check	Blue	small	
12	monotype	unchecked	unchecked	check	check	Yellow	ridiculouslyLarge	
13	brushScript	check	check	check	check	White	xxLarge	
14	brushScript	check	check	unchecked	check	Black	ridiculouslyLarge	
15	monotype	check	check	unchecked	check	Yellow	nominal	
16	arial	unchecked	check	unchecked	unchecked	Green	xLarge	
17	arial	unchecked	unchecked	unchecked	check	Black	nominal	
18	tahoma	check	unchecked	check	unchecked	Black	xxLarge	
19	tahoma	unchecked	check	check	check	Green	nominal	
20	tahoma	unchecked	check	check	check	Green	ridiculouslyLarge	
21	monotype	check	check	unchecked	check	Red	large	
22	arial	unchecked	unchecked	check	unchecked	Green	small	
23	brushScript	unchecked	check	check	check	White	xxLarge	
24	monotype	unchecked	unchecked	unchecked	unchecked	Green	xxLarge	
25	tahoma	check	unchecked	unchecked	unchecked	Blue	nominal	
26	arial	unchecked	check	unchecked	unchecked	Blue	large	
27	brushScript	unchecked	check	unchecked	unchecked	Blue	large	
28	arial	check	unchecked	unchecked	check	Yellow	xLarge	
29	tahoma	unchecked	check	check	unchecked	Blue	xLarge	
30	tahoma	unchecked	check	check	unchecked	Black	large	
31	arial	unchecked	check	unchecked	unchecked	Yellow	xxLarge	
32	arial	unchecked	check	unchecked	unchecked	Blue	xxLarge	
33	monotype	check	check	unchecked	check	White	ridiculouslyLarge	
34	tahoma	check	unchecked	check	unchecked	Yellow	xxLarge	
35	brushScript	check	check	unchecked	check	Red	small	
36	brushScript	check	check	unchecked	check	Green	xxLarge	
37	brushScript	unchecked	check	check	check	Red	xLarge	
38	tahoma	unchecked	unchecked	unchecked	unchecked	Yellow	small	
39	arial	check	unchecked	unchecked	unchecked	Blue	small	
40	monotype	check	check	unchecked	unchecked	Black	xxLarge	
41	arial	check	unchecked	unchecked	check	Red	xxLarge	
42	tahoma	unchecked	check	check	check	Black	xxLarge	
43	monotype	check	check	unchecked	unchecked	White	small	

Figure 4: Randomizing the n-wise combinations increases test coverage

the tools output.

Now, anytime the Monotype Corsive font is specified in a test defined by the output from the tool, either both bold and italic parameters will be checked or both will be unchecked, and only italic or bold and italic will be selected for Brush Script MT font. Being able to deal effectively with conditional or invariant constraints prevents false positives and eliminates massaging the output to just “make it work” or simply ignoring some tests altogether.

Another way to improve test effectiveness and increase the likelihood of testing with the “right” values is to increase the number of different combinations that are tested. An output randomization feature in the PICT tool enables the tool to generate different sets of n-wise combinations from the population of all possible combinations. Increasing the number of different n-wise combinations tested during the testing cycle can significantly enhance test coverage of different combinations and potentially expose new issues. Figure 4 compares

two different sets of pairwise combinations produced by the same input model file processed by the PICT tool.

Testing Highly Likely Variables “Enough”

Based on our experience, customer profiles, and other information, we might assert that some variables or combination of variables are used more frequently. There are several ways to increase the probability that important values are adequately covered in the tests. The easiest might be hard-coding important values. For example, if we needed to make absolutely sure that a size 12 font is tested, we can simply hard-code that variable in our model. Another way to improve the probability of testing highly likely variables is to weight important variables. Figure 5 (see the StickyNotes) illustrates how we modify the tool’s input file to give added weight to the nominal size, black, Arial font with the effects and styles unchecked.

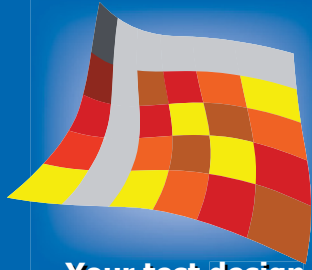
Weighted variables have a higher probability of being reused over lower or unweighted variables after all n-wise combinations are satisfied. But, a weighted variable does not automatically increase the frequency of that variable. So, to guarantee we test specific combinations, we can seed the PICT tool’s

input model with important combinations. For example, figure 6 (see the StickyNotes) shows a tab-delimited file used to seed the PICT output and ensure that this specific combination is always tested while still producing all other n-wise combinations.

A sub-model of important parameter combinations is another method to increase test coverage of important variable combinations. For example, if we need greater examination of the interaction between the font, size, and color variable combinations, we could bundle those parameters into a sub-model. The order of n-wise combinations of a sub-model is independent of the parent model for higher orders of combinations. Figure 7 (see the StickyNotes) illustrates a sub-model for the PICT input model for the font, size, and color parameters with an order of 2.

Sub-models increase the thoroughness of test coverage of certain parameter combinations, and this example increased the number of tests from approximately forty simple, pairwise

Testcover.com



Your test design requirements. Covered.™

What's the PROBLEM?

The world runs on software. Users demand new products faster than ever. We face tight development cycles and out-of-control complexity. Test tools and processes have not kept pace, causing:

- Product defects
- Service errors
- System failures
- Lost productivity
- Personal injury

SOLUTION: Testcover.com [Pairwise • SaaS • WSDL]

Our pairwise test case generator helps you get test results in a fraction of the time:

- Independent test design solution that's ideal whether you have three testers or 300
- WSDL interface integrates with current test tools
- No software to buy, install or maintain; full solution at **testcover.com**
- Used by corporations and universities around the world
- SAVE TIME AND MONEY on every project

Exclusive 30-day trial for Better Software readers

A 30-day trial of the world's best pairwise test case generator solution is at your fingertips. Visit **testcover.com** today and click on **BSM 30-DAY TRIAL**.

[Activation code: 66c1 34fc b9f7 23e2]



Testcover.com
+1 732 299 4852
mgmt@testcover.com

tests to more than eighty tests. Increasing the n-wise order of the sub-model to 3 further increased the number of combinatorial tests to more than 300! That's a lot of manual tests, but an automated, data-driven test case took only about thirty-five additional seconds to execute those 300+ tests, compared to the eighty tests identified in our pairwise-ordered sub-model in figure 7.

Good-enough Oracles

There are different techniques and approaches to testing that help us evaluate the various capabilities and attributes of a project and expose different categories of potential flaws. Regardless of the testing approach, we always need some type of oracle in order to determine the outcome of any test. There are different kinds of oracles, but the effectiveness of any oracle depends on having a clear purpose or goal for each test case and the ability to articulate the important conditions to validate specific test conditions, identify potential failures, or effectively deal with unexpected conditions. Automated oracles often may be more effective at evaluating discrete events and computational logic, while humans are perhaps better suited to analyzing behavioral aspects of the product from a system-level or end-user perspective. The key is to use the most relevant oracle in the appropriate context.

In our example, the purpose of our test case is to verify whether the various font attributes are correctly applied to the selected string in our text editor and that various combinations do not trigger unexpected system behavior. In this case, since we are not testing the system-level APIs that provide the different properties, we can use the underlying system-level APIs to get the font properties in our text editor and compare them against the expected result identified by our input variables for each combinatorial test.

Advancing the State of the Practice

It is important to emphasize that this and any other technique can be misused by an untrained tester, who simply tries to test all input variables without considering the common output condition affected by those input variables or who does not have a clear purpose for the test case. Simply put, when you use a tool or technique incorrectly, you usually end up with incorrect or less-than-satisfactory results.

But, used in the appropriate situation, this technique can help skilled testers systematically produce from all possible combinations a subset of tests that provides adequate test coverage and has a high probability of exposing a significant number of multi-modal faults caused by variable interaction. The bottom line is that, compared to other approaches, a trained tester using this technique with a customizable tool is the most effective approach known today for testing variable interaction of complex, interdependent parameters.

Of course, the real quandary of testing any complex system is that any set of tests is merely a small sample of tests from the virtually infinite population of all possible tests. So, any approach to testing is likely to exclude some decisive tests and fail to uncover some important bugs. As we pursue our practice of software testing to become more effective in our testing, we should gain a richer understanding of the entire system under test and learn to use a variety of techniques, tools, methods, and approaches in the appropriate situations. **{end}**

bj.rollison@microsoft.com

Sticky Notes

For more on the following topics go to www.StickyMinds.com/bettersoftware.

- References
- Figures 5-7

Attend Live, Instructor-led Classes Via Your Computer

*New Live
Virtual Courses
Now Available!*



NEW Live Virtual Courses:

- » Mastering Test Automation
- » Essential Test Management and Planning
- » Finding Ambiguities in Requirements
- » Getting Requirements Right the First Time
- » Testing Under Pressure

Live, instructor-led classes are now available right from your computer! SQE Training uses Cisco's WebEx technology to provide you with all the benefits and personal contact of classroom instruction right from your desktop. You get the same valuable content and instructor interaction as you would in the classroom but with the convenience and cost effectiveness of being online.

Live Virtual Package Includes:

- **Easy course access:** You attend training right from your computer, and communication is handled by a phone conference bridge utilizing Cisco's WebEx technology. That means you can access your training course quickly and easily and participate freely.
- **Live, expert instruction:** See and hear your instructor presenting the course materials and answering your questions in real-time.
- **Valuable course materials:** Our live virtual training uses the same valuable course materials as our classroom training. Students will have direct access to the course materials.
- **Hands-on exercises:** An essential component to any learning experience is applying what you have learned. Using the latest technology, your instructor can provide students with hands-on exercises, group activities, and breakout sessions.
- **Real-time communication:** Communicate real-time directly with the instructor. Ask questions, provide comments, and participate in the class discussions.
- **Peer interaction:** Networking with peers has always been a valuable part of any classroom training. Live virtual training gives you the opportunity to interact with and learn from the other attendees during breakout sessions, course lecture, and Q&A.
- **Convenient schedule:** Course instruction is divided into modules no longer than three hours per day. This schedule makes it easy for you to get the training you need without taking days out of the office and setting aside projects.
- **Small class size:** Live virtual courses are limited to the same small class sizes as our instructor-led training. This provides you with the opportunity for personal interaction with the instructor.



LICENSE TO OPEN SOURCE

A central graphic featuring the black silhouettes of two people, a man and a woman, facing each other in profile. They are positioned within a bright red circle. Both individuals have their hands raised, with their index fingers pointing upwards. The background of the entire page consists of concentric, wavy gray lines that create a sense of depth and movement.

BY KAMAL HASSIN AND KATHERINE CHIN QUEE

ISTOCKPHOTO

In 2009, the open source community witnessed a headline-catching lawsuit [1]: Best Buy, Samsung, JVC, and eleven other consumer electronics companies were named in a copyright infringement lawsuit—the scope of which is unprecedented.

The suit alleges the defendants violated the terms of the GNU General Public License (GPL) by distributing products containing a tool called BusyBox that combines many UNIX utilities into a single executable that is commonly incorporated into household electronic devices. In the lawsuit, the defendants were alleged to have not made open source code available to downstream users as they were required to do under the terms of the GPL license, resulting in potential damages, injunctive relief, and legal fees. BusyBox has also been at the center of several other high profile GPL violations that have been settled with Extreme Networks, Monsoon Multimedia, Xterasys Corporation, High-Gain Antennas, and Verizon.

These enforcement actions drive home the importance of taking inventory of what open source software is included in your products, knowing what licensing obligations apply to each open source component, and taking action to comply with these obligations. Open source license compliance is particularly important given the rapid growth of embedded computer systems, such as products—including Insignia Blu-ray Disc players and Samsung LCD HDTVs—identified in the BusyBox lawsuit against Best Buy.

What Is Open Source Software?

Open source software refers to software in which the source code is made available to the public. The open source community and various development ecosystems can modify, improve, and incorporate the code into other works. By contrast, proprietary software is usually distributed only as machine-readable, compiled code. Open source software is often referred to as free and, while most open source software is indeed free of charge, “free” actually refers to the freedom to use, modify, and redistribute the source code so long as other licensing obligations are met.

These licensing obligations are enabled by intellectual property (IP) rights which, in most legal jurisdictions such as the United States and Canada, ensure that software is automatically protected by copyright as soon as an original work has been created. Copyright law grants software owners the exclusive right to reproduce their software and to create additional software based on the original protected work, as well as the right to distribute their software. In general, open source licenses use these exclusive rights to create and reinforce an ecosystem that ensures that source code remains open and accessible so that successive developers can innovate around it. These open source ecosystems freely propagate, since anyone failing to fully participate in the ecosystem may be violating license obligations and could be held liable for copyright infringement. While license obligations are indeed part of the extended cost of using open source, there are also significant benefits, including productivity gains and reduced development expense relative to proprietary software. Additionally, as in the case of *Jacobsen v. Katzer* [2], courts have recognized the substantial benefits of distributing copyrighted works under public licenses. The court noted that program creators may generate market share for their programs by providing certain components free of charge. Similarly, a programmer or company may increase its national or international reputation by incubating open source projects.

What Are Open Source Licenses?

The Open Source Initiative (OSI) is a public benefit corporation that refers to itself as the steward of the open source definition (OSD). The OSI is a community-recognized body for reviewing and approving licenses as OSD-conformant. A license must comply with the OSI’s ten distribution terms in order to be approved as open source. The three major requirements include royalty-free redistribution, available source code, and the license must allow for modifications and derived works (works based upon the licensed work). Merely making the source code available does not necessarily mean the licensor has permitted you to modify or redistribute the software. In fact, the OSI has criticized companies that have advertised their software as open source when the license was not approved by the OSI board. Such software might be more accurately described as “source available” software.

More generally, license agreements are a type of contract. However, there are subtle differences between the interpretation of licenses and contracts that can have harsh implications for a party that is caught offside. For example, under a license it is easier to obtain an injunction when one’s property rights have been violated than it is for a breach of contract. An injunction orders a party to refrain from the infringing activity and can halt business operations, whereas the ordinary remedy for a breach of contract is damages.

Why Should I Care About Open Source Licenses?

Open source software is rapidly growing as companies that initially cast a wary eye on it now realize

its numerous benefits such as reduced cost and accelerated time to market. While open source offers many benefits, it also heightens the probability of code contamination from an IP perspective. Code contamination occurs when content is brought into a development project without regard for the licensing or copyright obligations.

Contaminated code can come back to haunt a company. In a merger, acquisition, or funding deal, uncertainty over IP ownership can generate risk, threaten successful closure, and affect software IP valuation and, consequently, overall business valuation. As evidenced by recent court cases [3], open source license violations can result in litigation that can drag on for years, draining company resources and producing negative press and public scrutiny.

What Are the Main Obligations of Open Source Licenses?

Despite the benefits of open source software, companies often shy away from it because they do not fully understand the obligations of various licenses. There is fear that using open source will require a company to give away all of its software for free—this is not accurate. Open source licenses can be diverse, ranging from permissive to restrictive.

In the StickyNotes, you'll find a matrix showing six of the most frequently used open source licenses along with the associated license obligations. Please note this matrix is for illustrative purposes only and is not an exhaustive list or explanation of open source licenses.

Lessons Learned

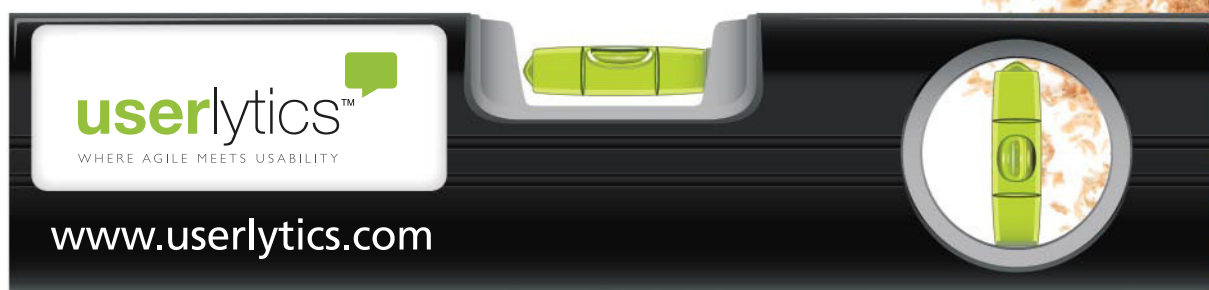
Many products commonly contain hundreds or even thousands of software components, and companies will increasingly be expected to account for the software IP ownership and obligations associated with those products. As seen in this article, licensors are willing to enforce the terms of their open source licenses in court. The lawsuits mentioned above demonstrate that open source users cannot simply ignore—knowingly or unknowingly—their licensing obligations without the risk of legal repercussions. The previous legal examples highlight the importance of understanding what open source software is, taking inventory of what open source is included in each product, identifying what licensing obligations apply, and ensuring compliance with these obligations. Fortunately, this process is now made easier with automated source-code-scanning tools that can analyze and identify the presence of open source code and what type of licensing applies. **{end}**

khassin@protecode.com
kchinquenee@protecode.com

Sticky Notes

For more on the following topics go to www.StickyMinds.com/bettersoftware.

- References
- The six most often-used open source licenses



The right tool to:

- reduce the time and cost of SW development
- improve the customer experience
- decrease QA and customer service spend
- increase the ROI of SW development

Contact Userlytics today: (415) 449-0502

Simple:

set up a usability test in just five minutes

Quick:

interpretable video-centric results in hours

Powerful:

fine-tune Design, Development, and QA

Experience the Advantages of Self-Paced eLearning



**Try our New Demos Now Available online
at www.sqetraining.com/eLearning**

New Courses Now Available in eLearning Format



eSoftware Tester Certification—Foundation Level *Become a Certified Software Tester from Your Desktop*

Are you looking for an internationally recognized certification in software testing? Delivered by top experts in the testing industry, eSoftware Tester Certification is an accredited training course to prepare you for the ISTQB™ Certified Tester—Foundation Level exam. This program is the only internationally accepted certification for software testing, accredited by the ISTQB™ through its network of national boards.



eMastering Test Design *The Art and Science of Creating Test Cases*

This course begins where many software testing courses end. Once the test plans are written, test teams are formed, and test tools are selected, it is time to create test cases. Since testing everything is impossible, the first step in test design is to choose a subset of all possible tests of program paths and data combinations to find important defects quickly.



eFoundation for Requirements Development and Management *A Roadmap to Success*

If you currently develop and manage requirements, manage people who do, or plan to do either in the future, this course is for you. This course teaches essential requirements development and management skills in a flexible self-paced eLearning format. The curriculum is a series of eight self-paced courses that build the foundation you need to successfully develop and manage requirements for business projects and software products.

Be trained from your desktop with SQE Training's eLearning courses. Experience classroom value with the convenience of self-paced instruction on the Web.

What are the benefits of eLearning?

- **Superior lesson content:** Developed and delivered by experts
- **90 Days access to all course materials:** Learners have unlimited 90 day access to online content
- **Access to expert test consultants:** Email questions and comments to the experts
- **Powerful multi-media format:** Students experience professionally narrated audio and video clips
- **Interactive exercises:** To reinforce new skills and concepts



www.SQETraining.com

The Roles of the Project Office in



VEER IMAGES

A project management office (PMO) that is engaged in and supportive of transitioning to Scrum can be a tremendous boon. Members of the PMO often view themselves as protectors and supporters of a practice, so a PMO can help implement and spread agile practices across the organization. However, when the PMO is not properly involved, it can be a source of resistance as it tries to defend the current process, rather than improve it.

The natural response of most people in the PMO is to resist the transition to Scrum, because much of the change is both personally and professionally frightening. Scrum scatters traditional project management responsibilities among the ScrumMaster, product owner, and the team, leaving project managers questioning their role. The absence of the PMO in most Scrum and agile literature adds to the natural concerns of PMO members.

In this article, we will ease those fears by looking at the type of work performed by PMOs in organizations that have successfully transitioned to Scrum. We will look at the contributions and work of the PMO in three areas: people, projects, and process.

People

Although it's called the project management office, the PMO has tremendous influence on the people involved in a Scrum transition. An agile PMO should do the following:

Develop a training program. There is much to adopting Scrum that will be new and unfamiliar to many team members. The PMO can be of tremendous assistance in putting together a training program, selecting outside trainers to deliver the training, or delivering the training themselves.

Provide coaching. Beyond training people, individual and small-group coaching is incredibly helpful. In a training class, the instructor says, "Here's how to do a sprint planning meeting," for example, and perhaps runs the class through an exercise to practice it. With coaching, someone with deep experience sits with the team and helps team members through their own real sprint planning meeting (or whatever skill is being coached). Early on, members of the PMO might not have these skills themselves, but they should focus on acquiring them from outside coaches and then do the hands-on coaching themselves.

Select and train coaches. A successful Scrum initiative will

Management Scrum

By Mike Cohn



eventually lead to more coaching needs than the PMO can manage on its own. Members of the PMO should identify and develop coaches by watching the teams they help and then providing training or assistance to help selected individuals become skilled coaches. These coaches usually retain their current jobs but are given additional responsibilities, such as spending up to five hours per week helping a specific team.

Challenge existing behaviors. When the organization begins to adopt Scrum, the members of the PMO look for teams who are falling back into old habits or whose old habits are preventing them from becoming agile. Later, members of the PMO can remind teams that Scrum is about continuous improvement and can help prevent the onset of complacency.

Projects

Although some project-oriented responsibilities go away with the change to an agile PMO, some responsibilities remain, including the following:

Assist with reporting. In most organizations large enough to have a PMO, there is usually something like a meeting or weekly report on the status of each project with the depart-

ment head. If this is a meeting, it should be attended by appropriate project personnel, such as the product owner or ScrumMaster. If it is a weekly, standardized status report, the PMO can assist in preparing the report.

Assist with compliance needs. Many projects need to comply with standards (ISO 9001, Sarbanes-Oxley, and so on) or with organization-specific rules, such as those for data security. An agile PMO can assist teams by making them aware of such needs, advising them on how to comply, and serving as a central clearinghouse for tips and shared knowledge on compliance and similar matters.

Manage the inflow of new projects. One of the most important responsibilities of an agile PMO is to assist in managing the rate at which new projects flow into the development organization. As described in chapter 10 of *Succeeding with Agile* [1], it is important to limit work to capacity. Otherwise, work piles up, leading to a litany of problems. For each project completed, a new project of the same size can be started. The agile PMO can serve as gatekeeper and help the organization resist the temptation to start projects too quickly.

Process

As keepers of the process, members of the PMO will work closely with the organization's ScrumMasters to make sure Scrum is implemented as well as it can be. These process-related activities include the following:

Assist in establishing and collecting metrics. As it did before becoming agile, the PMO can identify and collect metrics. Scrum teams are even more leery of metrics programs than traditional teams, so this is an area where the PMO should proceed cautiously. One thing an agile PMO should collect is information on how well teams are doing at delivering value.

Reduce waste. The PMO should aggressively help the team eliminate all wasteful activities and artifacts from its process. An agile PMO should avoid introducing documents, meetings, approvals, and so on unless absolutely necessary. It should also help teams look for things that they are doing that might not be adding value.

Help establish and support communities of practice. A community of practice is a group of liked-minded or like-skilled individuals. One of the most important things an agile PMO can do is to help encourage the formation of these communities and then support them after they begin. Not only do communities of practice help Scrum spread through the organization, they also help spread any good idea from one team to another.

Create an appropriate amount of consistency across teams. Most teams, especially Scrum ones, bristle at the thought of consistency enforced through dictate. The best type of consistency across teams comes from most or all teams agreeing that a particular practice is a good idea. The agile PMO facilitates this by making sure good ideas spread rapidly among teams. Two practices they can use for this are communities of practice and shared coaches.

Provide and maintain tools. In general, tool decisions should be left to individual teams whenever possible. On occasion, a community of practice might decide that there are sufficient benefits to choosing one tool for all projects. As a last resort, tool decisions can sometimes be made by the PMO, although this should be extremely rare. But, the agile PMO can assist teams by acquiring the appropriate tools and performing any configuration or customization necessary.

Coordinate teams. Because they work with individuals from many different teams, PMO members are vital in coordinating the work of separate teams. Someone from the PMO will often be the first to notice when the work of two teams starts to diverge or overlap. PMO members can provide value to teams by alerting teams to these situations when they occur.

Model the use of Scrum. Through their intensive exposure to Scrum, most agile PMOs quickly realize Scrum's usefulness as a general-purpose project management framework. At that point, many PMOs choose to use Scrum itself to run the PMO. They plan monthly sprints, conduct daily scrums, and so on just like any other team.

Work with other groups. The PMO can be of great assistance to teams in working with other groups, especially human resources and facilities. A PMO can help explain to the facilities group the negative impact of spreading a team across two floors or of having programmers and testers from

the same team sitting separately. A good PMO will work with human resources to remove from the annual review process questions that discourage teamwork.

Renaming the PMO

Many PMOs choose to rename themselves to better match their revised role. Though there is no one standard name, I have heard these most frequently: Scrum center of excellence, Scrum competence center, Scrum office, and development support.

Many people have become cynical and suspicious of name changes and of well-crafted names. That cynicism will be directed at the PMO if it is renamed but remains otherwise unchanged. So, whatever it's called, the PMO that supported the organization's sequential development process will need to change more than just its name to succeed with Scrum.

A PMO often has tremendous political clout and project experience. Though an adversarial relationship might work for a while, teams that have a PMO on board with Scrum not only avoid a possible source of resistance but also benefit from having a powerful ally. When possible, in transitioning to agile, choose to make friends rather than enemies. **{end}**

mike@mountaingoatsoftware.com

Sticky Notes

For more on the following topic go to www.StickyMinds.com/bettersoftware.

■ References

SALARY SURVEY

Better Software magazine
StickyMinds.com

2010

CALLING ALL SOFTWARE DEVELOPERS, TESTERS, AND MANAGERS

The 2010 Better Software magazine/StickyMinds.com Salary Survey is in full swing!

Don't miss your chance to contribute to the collective knowledge of industry salaries and employment trends.

Follow the link that best describes your employment level to answer a few questions.

The results will appear in the December 2010 issue of Better Software magazine.

STAFF LEVEL:

www.stickyminds.com/salariesurveystaff

MANAGEMENT LEVEL:

www.stickyminds.com/salariesurveymanagement

DIRECTOR LEVEL:

www.stickyminds.com/salariesurveydirector



NOVEMBER 14-19, 2010

ORLANDO, FLORIDA

AGILE DEVELOPMENT PRACTICES CONFERENCE

East

Conference Sponsor:



www.sqe.com/adpeast

- 100+ learning and networking sessions over six days
- In-depth pre-conference Training Classes and Tutorials
- Conference classes covering agile leadership, transitioning to agile, lean thinking, scaling agile, design, testing, and more
- Inspiring keynotes by Scott Ambler, David Hussman, Mary Poppendieck, and Niel Nickolaisen
- Agile Testing Workshop (two days)
- Free Sunday session: The Foundations of Agile Development
- Networking events: Receptions, bonus sessions, breaks, breakfasts, lunches, and more!
- Agile Leadership Summit

REGISTER BY OCTOBER 15, 2010 AND
SAVE UP TO \$200
GROUPS OF 2 SAVE EVEN MORE!



100% OF 2009 ATTENDEES RECOMMEND THE AGILE DEVELOPMENT PRACTICES CONFERENCE

BUILD YOUR CONFERENCE!

Conference schedule includes multi-day training classes, tutorials, keynote presentations, conference classes, Summit sessions, and more-packed with information covering the latest technologies, trends, and practices in agile development.



SUNDAY

Scrum Master Certification (3 days)
Practical Test-Driven Development (3 days)
Product Owner Certification (2 days)
The Foundations of Agile Development (Free bonus session)

MONDAY – TUESDAY

24 In-depth half- and full-day Tutorials
Multi-day training classes continue
Agile Testing Workshop



Popular Tutorials Include:

Agile Product Development: Building Meaningful Backlogs	Fostering Trust in Teams: A Leadership Practicum
Lean Software Development: Getting Started	Squeeze More Business Value from Your Agile Transition
SCRUM: A Manager's Guide	Using Metrics in Agile Environments
S.O.L.I.D. Principles of Systems Design	Successful Agile Requirements: Defining and Delivering Value
Writing Great User Stories	Release Planning: A Strategy for Success

WEDNESDAY – THURSDAY

4 Keynotes
36 Conference Classes
Networking EXPO
Special Events
...and More!



Covering topics like:	Leading Agile	Doing Agile	Agile Design
	Transitioning to Agile	Lean Thinking	Agile Testing
	Agile Project Management	Scaling Agile	And More
	Agile Requirements	Agile Products	

KEYNOTES BY INTERNATIONAL EXPERTS



Reality Over Rhetoric: Busting Some of the Myths Around Agile

Scott Ambler, IBM Rational



Are We There Yet? Challenges for the Next Decade

Mary Poppendieck, Poppendieck, LLC



Big Agility Requires Little-a agile

David Hussman, DevJam



Meaningful Metrics for Agile Teams and Organizations

Niel Nickolaisen, Headwaters, Inc.

"I am excited to take so many great ideas back to work with me. The facility was first-class. All of the speakers I had, tutorial and concurrent [sessions], were experienced in their fields and had valuable insight to share."

— Agile Development Practices Delegate

"I liked the ability to hear other views on agility—to [get a] better perspective of what others are doing."

— William Ploch, Project Leader, Energizer

FRIDAY

Agile Leadership Summit: Taking Agility to the Next Level

Join your peers and agile industry veterans to explore the unique challenges facing software development leaders as agile practices move into the mainstream. You'll hear what's working-and not working-and have the opportunity to share your experiences and successes.



Three Lenses-One Pair of Eyes

Jay Bourland, Pitney-Bowes



Leading Agile: The Big of It

Robert Begg, IBM



Unleashing Talent in Rapidly Changing Environments

Christine DelPrete, Amirsys



SAVE!

Ways to Save on Your Conference Registration

Special Early Bird Offer!

Receive up to \$200 off the regular conference registration fees if payment is received on or before October 15, 2010.

Bring a Buddy

Bring a colleague and save! Any two people registering at the same time save up to an additional \$200 off each registration

PowerPass Discount

PowerPass holders receive an additional \$100 off their registration fee.

Alumni Discount

Agile Development Practices alumni receive up to an additional \$200 discount off their registration fee.

Multi-day Training Classes + Conference

If you attend any of the training courses and conference, you save an additional \$500.

Groups of 3 or more Save 25%

Register a group of three or more at the same time and save 25% off each registration. To take advantage of this offer, please call the Client Support Group at 888.268.8770 or 904.278.0524 or email them at sqeinfo@sqe.com and reference promo code GRP3.

Check out all the Ways To Save at:
www.sqe.com/AgileDevPracticesEast/WaysToSave.aspx

Please Note: We will always provide the highest possible discount and allow you to use the two largest discounts that apply to your registration.



THE EXPO November 17-18

Visit Top Industry Providers Offering the Latest in Agile Development Solutions

Looking for answers? Explore the Agile Development Practices EXPO, designed to bring you the latest solutions in technologies, software, and tools covering all aspects of agile software development.

- Gather information to compare the latest solutions available in the industry
- Attend technical presentations and demos
- Meet one-on-one with some of today's most progressive and innovative organizations

For Sponsor/Exhibitor news and updates, visit www.sqe.com/adpeast.

Conference Sponsor:



Industry Sponsors:



Media Sponsors:



www.sqe.com/adpeast

COMBINE DISCOUNTS & SAVE UP TO \$400

TestComplete 8.0

BEVERLY, MA—SmartBear Software announces the new release of AutomatedQA TestComplete 8.0, which adds major new capabilities that boost productivity of individual users and further streamlines enterprise-level automated testing. Overall, more than 120 new features and enhancements have been added, driven by TestComplete's global community of thousands of loyal users.

Major features include:

- Test visualizer—TestComplete 8.0 automatically captures screenshots during test recording and playback. This enables quick and intuitive comparisons between expected and actual screens during test.
- New automation for data-driven tests—Creating data-driven tests is now highly automated using a new wizard to access data sources in databases, Excel, and CSV files. Users can create data-driven tests without scripting, which significantly improves test productivity.
- Support for new platforms and technologies—Users can now perform automated testing of native and managed applications built with Microsoft Visual Studio 2010 and the .NET Framework 4. TestComplete 8.0 also supports new Web technologies and platforms such as Silverlight 4 and Firefox 3.6.
- SoftwarePlanner and other defect tracking tools—TestComplete 8.0 is better integrated with SoftwarePlanner, SmartBear's development and test management software, featuring new capabilities to generate reports based on automated test runs and to automatically log defects. This version also introduces new integration with other defect-tracking tools—Atlassian JIRA and Axosoft OnTime—enabling users to post defects directly from the TestComplete user interface.

For more information, visit smartbear.com.

Hansoft 6.1

SWEDEN—Hansoft has released Hansoft 6.1. The new version includes extensive support for the lean development technique kanban, with the unique ability to handle multiple parallel swimlanes of workflow.

"Kanban is hot in the development community right now, and we are proud to introduce not only the most extensive and easy to use support for kanban but also a unique support for kanbans with multiple swimlanes of workflow. We discovered the importance of this when sitting down with demanding teams, looking at what was needed for kanban to really work in practice. Hansoft has the ability to combine lean and kanban with other methods such as Scrum and Gantt scheduling in the same project or program."

New features include:

- Kanban support in the wall view—the most extensive and easy to use for your team
- Ability to create kanbans with multiple parallel swim lanes. This unique Hansoft feature helps the most demanding teams simplify advanced processes.
- Extensive export-to-spreadsheet functionality exports all information from Hansoft to xls
- XML import and import of data into Hansoft with full support for custom columns for easy migration from other tools (other migration option tools also available)

For more information, visit hansoft.se/new-hansoft-6-1.

Copyright © 2010, Varad Corporation. All rights reserved.

S-C-A-L-A-B-L-E

Ská-lə-bəl adjective.
capable of being easily expanded
or upgraded on demand

Individual

Startup

Small group

Large Group

Big Business

Enterprise

UPGRADE OR DOWNGRADE AT ANY TIME

**ISN'T IT NICE TO KNOW
THAT BUGHOST SCALES
WHEN YOUR
BUSINESS NEEDS
CHANGE?**

- ✓ **BUG TRACKING**
- ✓ **REQUIREMENTS**
- ✓ **TEST CASES**
- ✓ **TASK TRACKING**
- ✓ **FREE TECH. SUPPORT**

**THE RIGHT TOOL.
THE RIGHT PRICE.
THE ORIGINAL**



www.BugHost.com

BugHost, the BugHost logo and "Seymour" the bug are trademarks of Varad Corporation.

FAQ

expert answers to
frequently asked
questions

by Rick Craig
rcraig@sqa.com

How can I become a better leader?

A good leader creates an environment that fosters self-motivation, which is largely done through establishing a common vision, providing the tools and training to facilitate success, and treating everyone with respect. How do you accomplish this? I rely on the Marine Corps Principles of Leadership, because this is what I know and they have proven to be successful. These principles are not unique to the Marine Corps. If you reflect on these principles, I think you will find that they are common sense and apply in almost every situation.

Know yourself and seek self-improvement: If you are learning, it will encourage your staff to do the same.

Be technically and tactically proficient: You don't have to be an expert in every facet of testing, but you should understand the tools of the trade.

Develop a sense of responsibility among your staff: Can you leave your staff for a day or week and know that things will continue to function smoothly?

Make sound and timely decisions: I am not encouraging rash decisions, but delaying a decision can lead to inefficiency and potential morale problems.

Know your staff members and look out for their welfare: Is your staff underpaid? Overworked? Lacking respect? Addressing issues like these head on will motivate your staff even if you are unsuccessful in resolving the issues.

Keep your staff informed: Reading the staff into what is going on and the rationale behind it is a sign of trust and respect and will help prevent rumors and morale problems.

Take responsibility for your actions and the actions of your staff: Ultimately, if you are the leader, you will get "credit" for whatever your staff has done—good or bad. Give credit to your staff members for their contributions (your reward is the success of the mission) and accept responsibility for failures.

Ensure that tasks are understood, supervised, and accomplished: Good leaders will express what they want and not how to do it. Allowing staff members to determine how to accomplish a task creates a motivating atmosphere and encourages innovation.

Train as a team: Learn common methods and terms, and establish esprit de corps.

Employ your staff in accordance with its capabilities: If your team's business acumen is conducting black-box testing, abandoning that expertise to pursue a path of only doing structural testing would not necessarily be a wise choice.

Set a good example: If you speak in one manner and behave in another, the rest of these principles do not matter.

In addition to the Marine Corps Principles of Leadership outlined above, I would like to remind would-be leaders to establish high expectations of your staff but not to seek perfection. The so-called zero-defect mentality inevitably squashes morale and stifles innovation. Good luck.

Product Owners Should Care About Quality

Software quality influences a product's success, and the product owner is responsible for that success.

by **Roman Pichler** | roman.pichler@romanpichler.com

Product owners are responsible for product success. As a member of the Scrum team, the product owner collaborates with the ScrumMaster and team, as well as customers, users, and other stakeholders, to create a great product. Unfortunately, product owners often focus on delivering functionality and tend to neglect quality—rather than viewing quality as an enabler to deliver new features faster and to create lasting value.

Why Should Quality Matter to Product Owners?

Software quality significantly impacts product success: It influences customer satisfaction and brand equity, impacts the total cost of ownership and life expectancy of the product, and determines the product's competitiveness. Therefore, quality matters; it plays a key role in creating great products, as I explain in more detail below.

Customers will only be satisfied if the product's functionality works reliably and as expected. Defective software not only leaves customers dissatisfied, frustrated, or angry, but it also damages brand equity. Think about the issues Microsoft experienced with Windows Vista, including performance and interoperability problems. As a consequence, the company discontinued the Vista brand, calling its next operating system Windows 7. Only when quality meets functionality is true value created.

Quickly releasing poor-quality software may achieve a short-term win, but it incurs technical debt—the software becomes difficult to extend and maintain. [1] This results in high development cost and long lead time for new functionality. Software with inadequate quality often has to be replaced sooner rather than later, resulting in a short life expectancy and a poor return on investment.

Getting quality right is the prerequisite to leveraging customer feedback on early product increments, swiftly releasing software in response to the latest market development, and quickly bringing new functionality to the market. Compromising software quality means trading short-term gains for

longer-term growth, cheating the company of a better, brighter future.

What Can Product Owners Do to Ensure Quality?

As quality impacts product success, it should be a concern for the product owner as the individual who is first and foremost responsible for product success. To help get the quality right, product owners should apply the following recommendations:

Think products, not projects. A product owner should be responsible for a product for an extended period of time and ultimately manage the product life-cycle. Embracing a product perspective means viewing a project as a means to an end—the project simply brings the next product version to life. Longer-term responsibility counteracts the temptation to compromise quality in order to finish the current project fast and get the next release out as soon as possible. “Thinking products, not projects” creates a desire for sustained success and encourages long-term thinking.

Create a common understanding of quality. Make sure that a definition of “done” is available for each product

and apply that definition properly. The definition should clearly state the general quality criteria that product increments must fulfill. “Done” usually requires that a (potentially) shippable product—executable software that has been tested and documented and could be released—is available at the end of each sprint. As a consequence, quality assurance and control measures form an integral part of the development work instead of being carried out at the end of the project as an afterthought. Be specific regarding what “tested” and “documented” mean for your product. Some teams with which I have worked used metrics to refine and measure the quality criteria. As the product owner, you have to apply the done criteria to accept or reject work results when reviewing items—only work results that fulfill all the done criteria can be accepted. Enforcing the definition of done makes the product owner the guardian of quality.

“Compromising software quality means trading short-term gains for longer-term growth, cheating the company of a better, brighter future.”

Minimize defects and unnecessary rework. Together with the team, regularly groom the product backlog, and be available to answer questions as they arise. Due to its dynamic nature, the product backlog needs continued attention and care. New items emerge; existing ones are adjusted or removed. Items must be prioritized and estimated, and the high-priority items have to be detailed for the next sprint planning meeting. For example, user stories that are likely to be implemented in the next sprint should now be small enough and have well-formed acceptance criteria. Jointly grooming the product backlog ensures that it contains the right items in the right order, and it increases the likelihood that a story's implementation will meet the product owner's expectation. When it comes to product owner availability, I often suggest the one-hour rule: Product owners should spend on average at least one hour per day with their teams. This ensures that the product owner is available to quickly answer questions and provide early feedback on work results.

Invest in quality. Accept that team members need time to create high-quality software. Regard agile development practices such as test-driven development and continuous integration as essential to ensure sustained product health. Allowing team members time to refactor the software or to experiment

with new practices and tools may result in a lower velocity in the short term, but it speeds up development in the mid to long term.

Summary

Quality should be a concern for everyone on the Scrum team—the product owner, the ScrumMaster, and the team members. Since quality influences product success, it is in the product owner's best interest to care about quality and to collaborate with the team to ensure that a product with the right quality is created. Neglecting quality may achieve short-term gains, but it wastes longer-term growth and a better, brighter future. **{end}**

Sticky Notes

For more on the following topics go to www.StickyMinds.com/bettersoftware.

- References
- Further reading

index to advertisers

ADP East 2010	www.sqe.com/adpeast	29–32
BugHost	www.BugHost.com	33
Hewlett-Packard	www.hp.com/go/agile	Back Cover
Jama Software	www.jamasoftware.com/go	13
Microsoft	www.microsoft.com/visualstudio/test	6
MindTree	www.MindTree.com	Opposite Page 10
MindTree	www.MindTree.com	15
Rally Software	www.rallydev.com/bsm	Inside Front Cover
Ranorex	www.ranorex.com	5
Seapine	www.seapine.com	1
SQE Training—eLearning	www.sqetraining.com/eLearning	25
SQE Training—Live Virtual Training	www.sqetraining.com/Virtual Training	21
STAREAST 2011	www.sqe.com/STAREAST	Inside Back Cover
TechExcel	www.techexcel.com	2
Telerik	www.telerik.com/automated-testing-tools	10
TestCover	www.testcover.com	20
TotalView	www.totalview.com	14
Userlytics	www.userlytics.com	24

Display Advertising advertisingsales@sqe.com

All Other Inquiries info@bettersoftware.com

Better Software (USPS: 019-578, ISSN: 1553-1929) is published six times per year January/February, March/April, May/June, July/August, September/October, November/December. Subscription rate is US \$40.00 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 or call 800.450.7854. Back issues may be purchased for \$15 per issue (plus shipping). Volume discounts available. Entire contents © 2010 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.

SOFTWARE TESTING ANALYSIS & REVIEW

2011

**STAR
EAST**

THE GREATEST SOFTWARE TESTING CONFERENCE IN THE UNIVERSE (WE THINK!)

**Mark Your
Calendar Now!**

MAY 1-6, 2011
ORLANDO, FLORIDA
ROSEN SHINGLE CREEK



www.sqe.com/stareast



HARNES



the power of Agile.

HP Application Lifecycle Management for Agile initiatives

Looking to improve the visibility and consistency of your Agile initiatives?

The HP Application Lifecycle Management solution goes beyond traditional development management. It's the Agile solution that delivers quality *and* value, speed *and* control, for the whole team *and* the whole lifecycle.

Outcomes that matter.

Get more information at
hp.com/go/agile

