



SOFTWARE TESTING

ANALYSIS & REVIEW

The Greatest Software Testing Conference on Earth

**STAR
EAST**

ORLANDO, MAY 4-8, 2009
FLORIDA

www.sqe.com/stareast

REGISTER EARLY AND SAVE UP TO \$300!



- 34 In-depth pre-conference tutorials on how to be more efficient and effective
- 40 Concurrent sessions on how to turn challenges into opportunities
- 5 Keynote presentations from top testing experts with experience in down times
- Networking opportunities to see how others are handling economic challenges
- Largest Testing EXPO event to help you find solutions
- Testing & Quality Leadership Summit on Friday to help you become a leader in the quality movement

99% OF 2008 ATTENDEES RECOMMEND STAREAST TO OTHERS IN THE INDUSTRY



New for 2009!

Testing & Quality Leadership Summit

March 2009

\$9.95 www.StickyMinds.com

BETTER SOFTWARE

BEEN THERE,
DONE THAT
Learning from
old code

OFF THE BEATEN PATH
Discover freestyle
exploratory testing

The Print Companion to  **StickyMinds.com**

Lean Portfolio Management Guiding IT Projects with Business Value

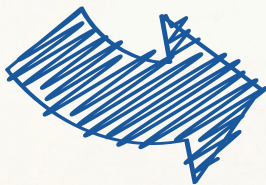
FIND THE BUG INSIDE & WIN
AN AMAZON GIFT CARD!

Rally Customers are 50% Faster to Market ...And 25% More Productive



"As compared to industry averages, Rally customers are 50% faster delivering their software to market and increased their teams' productivity by 25% while maintaining normal defect counts."

—QSM Associates



Cut costs with Rally and Agile
go to www.rallydev.com/agilevalve/bsm
Get started now!



Scaling Software Agility

www.seapine.com/bettertcm

Satisfy your quality obsession.



TestTrack® TCM
Test Case Management

QA Wizard® Pro
Automated Testing

Seapine CM®
Change Management

Surround SCM®
Configuration Management

TestTrack® Studio
Test Planning & Tracking

TestTrack® TCM
Test Case Management

TestTrack® Pro
Issue Management



© 2009 Seapine Software, Inc. All rights reserved.

Full-Time Quality Assurance Manager—Immediate Opening

Don't work yourself to death. Use TestTrack® TCM to manage your testing effort.

TestTrack TCM puts you in control of test case planning and tracking, providing better visibility over the testing effort and giving you more time to manage your team. With TestTrack TCM your team can write and manage thousands of test cases, select sets of tests to run against builds, and process the pass/fail results using your development workflow.

- Know instantly which test cases have been executed, what your coverage is, and how much testing remains.
- Manage suites of platform-specific compliance tests, functional tests, and performance tests in one central location.
- Assign tests to your QA team, track results, and report on performance and workload.
- Use test variants to target multiple platforms with the same test case for more efficient test case management.
- Streamline the QA > Fix > Re-test cycle by pushing test failures immediately into the defect management workflow.
- Achieve complete traceability between test cases and defects with seamless TestTrack Pro integration.
- Ensure all steps are executed, and in the same order, for more consistent testing.

 **Seapine Software™**

Perforce | The *Fast* Software Configuration Management System



Perforce Technical Support Fast Turnarounds. Precise Answers.

Our global support teams are always available to share their expertise in person – no scripted responses, answering services, or dispatch centers. At Perforce Software, our highly experienced technical support engineers take pride in providing fast turnarounds with precise answers.

Keeping your projects on track requires a support team that is ready to help right when you need it. You can count on Perforce's Fast SCM System and legendary technical support to give you the winning advantage.

PERFORCE
SOFTWARE

Download a free copy of Perforce,
no questions asked, from www.perforce.com. Free technical
support is available throughout your evaluation.

All trademarks and registered trademarks are property of their respective owners.

Cover Story

LEAN PORTFOLIO MANAGEMENT 26

Improving your software development process is only valuable if it fills the highest priority needs for your business clients with speed and quality. Lean principles provide guidance on how to create a structure that lets business priorities drive the selection of the right products for creation and enhancement. *by Guy Beaver*

Features



Columns & Departments

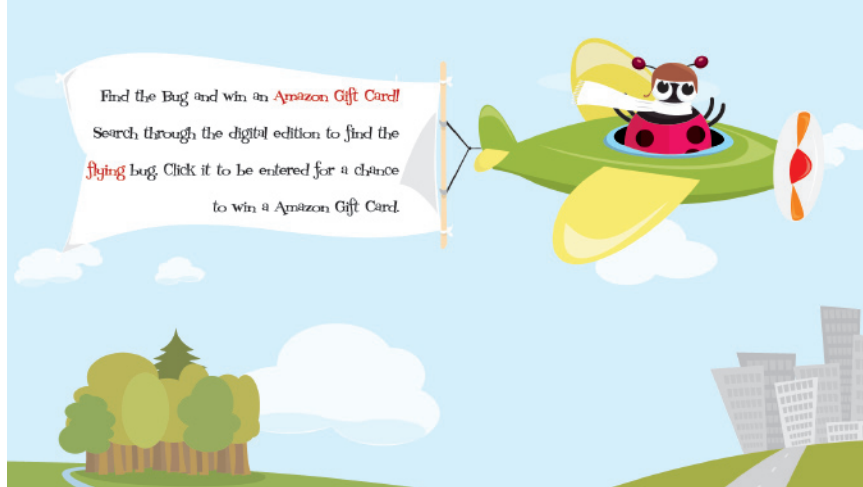
In Every Issue

- Mark Your Calendar **4**
- Contributors **6**
- Editor's Note **8**
- eLightenment **13**
- Product Announcements **51**
- 10 Things You Might Not Know About ... **54**
- Ad Index **56**

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

Subscribe today to get ten issues.

Visit www.BetterSoftware.com
or call 800.450.7854.



TAMING THE HEADLESS BEAST 32

The benefits of Web services are becoming widely demonstrated and accepted. However, these benefits are not without their own challenges. How can you enter data and verify the response of a system without a GUI? Are you ready to tame this headless beast? *by David Fern*

BUILDING A FOUNDATION FOR STRUCTURED REQUIREMENTS 40

Aspect-oriented requirements engineering (AORE) is a new methodology that can help us further improve the analysis, structure, and cost of development of software requirements. The second part of this two-part series focuses on the AORE specification techniques. *by Yuri Chernak*

TECHNICALLY SPEAKING 11

The Missing Measurement • *by Lee Copeland*

In these times, many of us are being told to “do more with less.” A more useful approach is “invest our organization’s scarce resources where the return is the greatest.”

CODE CRAFT 18

The Ghost of a Codebase Past • *by Pete Goodliffe*

Revisiting your old code can be an enlightening experience. Look back at your old code to see how your technique has improved, how your programming skills have progressed, and what you can learn from it.

TEST CONNECTION 20

Off the Trails • *by Michael Bolton*

A focused approach toward testing a product is important, but sometimes we discover information that we didn’t anticipate at all. If we vary our approaches, we might find something surprising and broaden our coverage.

MANAGEMENT CHRONICLES 22

Go, Team! • *by Patrick Bailey*

Fed up with good ol’ boy salesmen, a manufacturing mind-set, just “get-it-out-the-door” directions? A little assertiveness, a few ounces of patience, a dash of charm, a lot of leadership, and some attitude adjustment by everyone might help.

THE LAST WORD 55

Reloadable Test Data-O-Matic • *by Tanya Dumaresq*

Reloadable test data takes more time up front (as compared to on-the-fly data creation), but saves blood, sweat, and tears in the long term. It also virtually eliminates “works on my machine” bugs, creates a more intricate and realistic environment, and is the first step on the road to test automation.

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

MARK YOUR CALENDAR

TRAINING WEEKS

www.sqetraining.com/public

Testing

March 23–27, 2009

Boston, MA

April 20–24, 2009

San Diego, CA

Agile Software Development

March 30–April 3, 2009

Washington, DC

April 20–24, 2009

San Francisco, CA

SOFTWARE TESTING CERTIFICATION

www.sqetraining.com/certification

March 17–19, 2009

Nashville, TN and Vienna, VA

March 31–April 2, 2009

Boise, ID

April 14–16, 2009

San Francisco, CA and New Jersey

CONFERENCES

STAREAST 2009

Software Testing Analysis & Review

www.sqe.com/stareast

May 4–8, 2009

The Rosen Centre Hotel

Orlando, FL

Better Software Conference & EXPO 2009

www.sqe.com/bettersoftwareconf

June 8–11, 2009

The Venetian Resort

Las Vegas, NV

BETTER SOFTWARE

Publisher

Wayne Middleton

Vice President of Publishing

Holly N. Bourquin

Editor in Chief

Heather Shanholtzer

Editorial

Managing Technical Editor

Lee Copeland

Editor, StickyMinds.com

Francesca Matteu

Managing Editor, Multimedia

Joseph McAllister

Production Coordinator

Cheryl M. Burke

Design

Creative Director

Catherine J. Clinger

Advertising

Senior Advertising Sales Manager

Shae Young

Advertising Sales Manager

Joe Anderson

Production Coordinator

April Evans

Circulation and Marketing

Circulation Coordinator

Jamie Green-Gago

Marketing Coordinator

Sidney White

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



Take quality to the next level



DevTest Studio

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

DevTrack

Use DevTrack to track defects/issues

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

DevTest

Use DevTest to manage your testing

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

TestLink

Use TestLink to automate your testing

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

TechExcel

www.techexcel.com | 1-800-439-7782



GUY BEAVER is vice president of enterprise engagements and senior consultant at Net Objectives. He has more than twenty-four years of experience in IT and software delivery focused in financial services and aerospace industries. Guy began his career as a developer and served as delivery manager for multimillion dollar projects. He has managed IT organizations as director and CIO. Guy trains and consults business and technical teams with a practical application of lean principles that allows agile and Scrum teams to scale and integrate while remaining tightly coupled with high ROI activities.



PATRICK BAILEY has more than twenty years of experience in software development. He now teaches full time at Calvin College in Grand Rapids, Michigan. Contact Patrick at pmb4@calvin.edu.



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



YURI CHERNAK, PH.D., is the president and principal consultant of Valley Forge Consulting, Inc. Yuri has worked for a number of major financial firms in New York, leading QA governance committees in IT and helping clients improve their software requirements and software testing practices. Yuri is a pioneer in implementing a new discipline—aspect-oriented requirements engineering—for financial applications on Wall Street. He is a member of the IEEE Computer Society, has been a speaker at several international conferences in the US and Canada, and has published papers in the IEEE publications and other professional journals. Contact Yuri at ychernak@yahoo.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. Lee 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.



As Macadamian's QA team leader, **TANYA DUMARESQ** has been providing quality assurance for companies like FileMaker, Entrust, and March Networks since 2001. Tanya has presented at STARWEST and STAREAST, and has conducted training sessions for customers in three countries. In her spare time, she kicks butt on the Macadamian softball team and plays a mean foosball game.



DAVID FERN is a seasoned software testing professional with more than ten years of experience breaking software in the public and private industry. Currently he is an automation test engineer and software testing technical lead at the Social Security Administration. David has developed and implemented software testing strategies for large government Internet application deployments as well as enterprise-wide point-of-sale systems in the private sector. He has trained and mentored teams on processes and strategies for testing new technologies such as Web services and has had multiple articles and techniques published. In 2008, David spoke at the STAREAST conference and at HP Universe. He is ISTQB/ASTQB certified at the tester-foundation level, a Mercury Interactive Quick Test Professional 9.0 Specialist, and a Microsoft MCSE and MCDBA.



PETE GOODLIFFE is a software developer, columnist, speaker, and author who never stays in the same place in the software food chain. His projects range from OS implementation, through audio codecs, to multimedia applications. Pete's popular book, *Code Craft: The Practice of Writing Excellent Code*, is a practical and entertaining investigation of the entire programming pursuit—in about 600 pages. No mean feat! He has a passion for curry and doesn't wear shoes.



JEFFERY PAYNE is CEO and founder of Coveros, Inc., where he has led the startup and growth of the company. Prior to Coveros, Jeff was chairman of the board, CEO, and co-founder of Cigital, Inc. Under his direction, Cigital became a leader in software security and software quality solutions, helping clients mitigate the business risks associated with failed software. Jeff is a recognized software expert and speaks to companies nationwide about the business risks of software failure. He has been a keynote and featured speaker at CIO and business technology conferences and frequently testifies before Congress on issues of national importance, including intellectual property rights, cyber-terrorism, and software quality. You can reach Jeff at jeff.payne@coveros.com.



THE “MORE WITH LESS” MANTRA

Recently, the employees of Software Quality Engineering (publisher of *Better Software* magazine) were given the opportunity to attend a thirteen-week money management class. In this class, we are learning how to take control of our personal finances by getting out of debt, saving money, and investing for the future.

This course is being offered at a good time, given the shaky state of the economy. Not only is it helping us get control of our personal financial lives, it also makes good business sense. Armed with the knowledge of how to make the most of our money and put it to work for us—rather than feel like we are at the mercy of our finances—we are better able to concentrate on *work* at work and be more productive.

With companies trying to find ways to make their money go further, the “do more with less” refrain is being repeated in workplaces around the world. But how does an organization actually begin to adopt this as a practice without sacrificing quality?

In this month's issue, you will find several articles that can help you get started.

In his Technically Speaking column, “The Missing Measurement,” Lee Copeland suggests that while many of us are being told to “do more with less,” a more useful approach is to “invest our organization's scarce resources where the return is the greatest.” To do so, he says, we must define the desired financial benefits along with the requirements when developing a system.

In this issue's cover story, “Lean Portfolio Management,” Guy Beaver discusses how improving your software development process is only valuable if it fills the highest-priority needs for your business clients with speed and quality. Lean principles provide guidance on how to create a structure that lets business priorities drive the selection of the right products for creation and enhancement. By managing your prioritized business features in a visible portfolio, you can hasten delivery of value to your customers.

And in “Building a Foundation for Structured Requirements,” Yuri Chernak wraps up his discussion of aspect-oriented requirements engineering (AORE) that he began in the January/February 2009 issue. AORE is a new methodology that can help us improve the analysis, structure, and cost of software requirements development.

As always, I hope you enjoy this issue of *Better Software* magazine. Drop me a note to let me know how you've put this issue to work for you.

Happy Reading!

Heather Shanholtzer
HShanholtzer@sqe.com

BETTER SOFTWARE

CONFERENCE & EXPO

JUNE 8-12, 2009
LAS VEGAS, NEVADA
THE VENETIAN

Conference Sponsor:



www.sqe.com/bsce



KEYNOTES BY INTERNATIONAL EXPERTS



Tim Lister
Atlantic Systems Guild, Inc.



Andy Kaufman
*Institute for Leadership
Excellence & Development, Inc.*



Michele Sliger
Sliger Consulting, Inc.



Jonathan Kohl
Kohl Concepts, Inc.

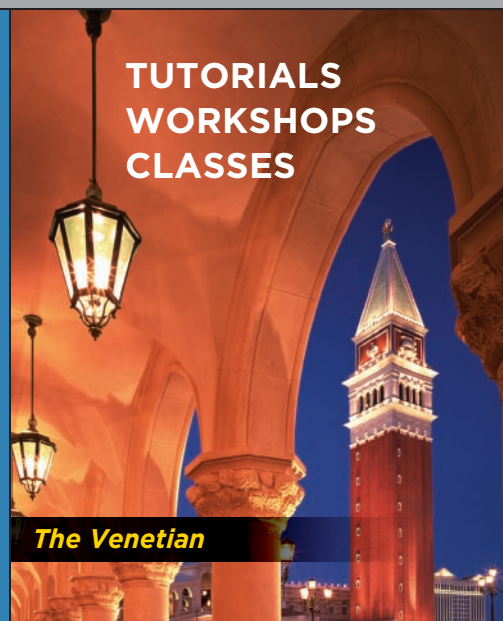


NEW FOR 2009!
a co-located event

Agile Leadership Summit
Friday, June 12, 2009



TUTORIALS WORKSHOPS CLASSES



The Venetian



ORLANDO
GAYLORD PALMS
MARCH 16-18
2 0 0 9

PAYMENT:
**SCRUM ALLIANCE
& PMI MEMBERS
\$1,500**
**NON-MEMBERS
\$1,800**

**LIMITED
SPACE!**

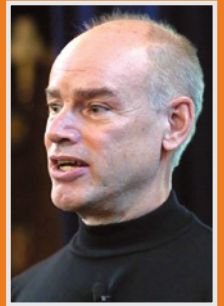
The Last Two
Scrum Gatherings
Sold Out Weeks
In Advance!

DON'T MISS THIS ONE:

TOP HEADLINERS IN AGILE & PMI TO PARTICIPATE IN THE ORLANDO SCRUM GATHERING

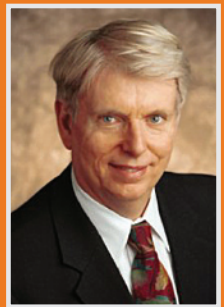
Ken Schwaber, Control Chaos, Scrum Co-Founder

Ken Schwaber is a Scrum evangelist. Ken developed Scrum with Jeff Sutherland in the early 1990's and is one of the authors of the Agile Manifesto. He was a founder and now President of the Scrum Alliance. Ken has been in software development for over 30 years. He has written three books about Scrum: Agile Software Development with Scrum, Agile Project Management with Scrum, and The Enterprise and Scrum.



Jeff Sutherland, Scrum Training Institute & Scrum, Inc., Scrum Co-Founder

Jeff Sutherland started the first Scrum at Easel Corporation in 1993. He worked with Ken Schwaber to emerge Scrum as a formal process at OOPSLA '95. Together, they extended and enhanced Scrum at many software companies and IT organizations and helped to write the Agile Manifesto. You can learn from Jeff's experience as consultant to the world's leading companies.



Headliners and featured speakers include:

Gregory Balestrero, President and CEO of PMI

Alistair Cockburn, among the founders/leaders of the Agile Movement

Mike Cohn, Mountain Goat Software

Jim "Cope" Coplien, father of Organizational Patterns

Ron Jeffries, XProgramming.com

Plus many more industry speakers, authors, trainers and innovators.

Register Today at www.scrumalliance.org



The Missing Measurement

by Lee Copeland

When an organization purchases new equipment, it expects that investment will, in some way, improve its products, services, productivity, or quality. If the purchase cost is significant, the organization will first compute the expected return on investment to quantify the monetary gain that will be achieved. If this gain does not exceed a specific threshold (for example, the return that could be achieved simply by investing the cost of the equipment in an interest-bearing account), then the purchase is not economically justified and probably should not be made.

During the life of most software projects, we measure and track schedule, cost, benefits, and quality. These measurements help direct our efforts as the project moves forward toward completion. However, after the project is completed—with the software installed, the most immediate defects removed, and the users busy at their keyboards—we may and, in fact, should ask, “Was this a good investment of our organization’s limited resources? Did we achieve an acceptable return on our investment?”

Unfortunately, many organizations cannot answer this question because they failed to specify the business requirements of the product before the project began. Most organizations define functional and non-functional requirements, user requirements, and business rules. These are vital because they define the product to be created and allow us later to evaluate the product against these specifications. But, these types of requirements say nothing about whether the organization’s investment was a good financial decision.

Business requirements define the high-level business objectives of the organization. They describe the financial, marketplace, and other benefits sought through developing a system and often are phrased in monetary terms (profit, rate of return, market share, etc.). Examples include: Capture 80 percent market

share within two years, limit support costs to 1 percent of revenue, receive no more than one complaint for every 10,000 transactions, make 22 percent pre-tax profit. Note that none of these business requirements says anything about the product or the project that will create it. They will, however, let us later evaluate the success of this product from a business point of view.

The place to document these business requirements is in the *product vision*, a document that becomes the foundation of all subsequent project work. This document has other names including the project charter, business case document, and marketing requirements plan. Whatever its name, it defines a vision of the product, the business requirements, and the business context including opportunities, risks, scope, and constraints.

One way to define the product succinctly is to use Geoffrey Moore’s template from *Crossing the Chasm*:

For [target customer]
Who [statement of need or opportunity]
The [product name]
Is [a product category]
That [key benefit, compelling reason to buy or use]
Unlike [primary alternative]
Our product [statement of primary differentiation and advantages of new product].

When completed, this simple statement will define a high-level vision that will both guide and bound further product development.

Unfortunately, in our rush to start projects, many organizations skip the product vision and jump straight into defining product requirements. For example, in his book *Agile Modeling*,

“...these types of requirements say nothing about whether the organization’s investment was a good financial decision.”

Scott Ambler suggests projects start with an initial requirements up front (IRUF) phase to define the high-level requirements represented in user stories or use cases and the scope and boundary of the product represented in a context diagram. Ambler writes that a key goal of the IRUF

phase is to get out of it as quickly as possible and get on to the “real work.” Note the lack of any thought of defining business requirements. Ambler is not alone in this approach—most agilists ignore defining business requirements that would later make it possible to evaluate whether an organization made good choices in asset allocation.

To my knowledge, in the agile community, only Jim Highsmith writes about the value of the product vision. In *Agile Project Management*, he states, “While the details of requirements and design can be volatile and fuzzy, the overarching business goal and product vision must be clear. Absent a clear vision, the exploratory nature of an agile project will cause it to spin off into endless experimentation. A clear vision must bound exploration.”

In these times, many of us are being told to “do more with less.” Often, this meaningless phrase is simply thrown out to “motivate” people who are already making the best of a bad situation. A more useful phrase would be “invest our organization’s scarce resources where the return is the greatest.” That phrase is both meaningful and actionable. Creating better software requires us to make better business decisions, and those should be based on better business requirements. The product vision is the place to start. **{end}**

SOFTWARE TESTING TRAINING

*Accelerate Your Career
& Empower Your Team*

NEW

**SPRING 2009
SCHEDULE**

**BUILD-YOUR-OWN
TRAINING WEEK**

Maximize the impact of your training by pairing two courses in one week and save up to \$500. Receive a \$50 Amazon Gift Certificate for registering 6 weeks prior to training. For a complete list of courses available, visit www.sqetraining.com or call 888.268.8770 or 904.278.0524.



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

TRAINING WEEK LOCATIONS

March 23-27, 2009	<i>Boston, MA</i>
April 20-24, 2009	<i>San Diego, CA</i>
June 1-5, 2009	<i>Chicago, IL</i>



The following courses are PMI Certified:

- ISTQB Software Tester Certification—Foundation Level
- Systematic Software Testing
- Mastering Test Design
- Test Management

Choose from 11 specialized training courses:

THREE-DAY COURSES (Monday - Wednesday)

- Systematic Software Testing
- ISTQB Software Tester Certification—Foundation Level
- Test Management
- Writing Testable Requirements
- Just-in-Time Software Testing

TWO-DAY COURSES (Thursday - Friday)

- Mastering Test Design
- Performance, Load, and Stress Testing
- Lean-Agile Testing Practices
- Requirements-Based Testing
- Exploratory Testing in Practice

ONE-DAY COURSE (Thursday)

- Test Process Improvement*

**Offered only in Boston and Chicago.*



Let SQE Training come to you. For more information about on-site training courses, contact SQE Training at 888.268.8770 or 904.278.0524 x212 or 233 or email onsitetraining@sqe.com.

EDITOR'S PICK

A Fire-Free Work Zone

I love my secret and not-so-secret addictions. There's a controllable addiction to exercise, an insatiable addiction to salsa and all that is spicy, a sweet-tooth addiction to southern-style iced tea, and a love for extreme activities like skydiving. I absolutely love conquering challenging, even dangerous, activities. That is why I'm the first to hop in a bungee chair at a run-down carnival. That's why the first time I ever went backpacking, I traveled alone—just to prove I could do it. That's why in my younger days, I raced cars with my friends. The part of me that is a slight adrenaline addict has caused me to become better at responding to events rather than preparing for or even avoiding them.

This has not proven extremely beneficial at work. Reaction is my forte, not pro-action. Knowing this, I set out a year ago to be more pro-active. It's been a long and difficult journey for this adrenaline addict. Forcing myself to stop before reacting takes a lot of energy, but it's important to recognize what causes problems in production so you can avoid bigger problems. The result has been a slow awakening to behavioral patterns that are likely to cause problematic work. Mistakes tend to happen when one rushes to stay ahead of work or on schedule, when one creates a new process or streamlines an old one without first consulting the rest of the team, and especially when people rush through work because they're so behind. This list isn't complete—not by a long shot—and a small list of identifiable precursors to problems doesn't present the solution. But Linda Hayes has written a gem of an article about this very subject.

Linda says that people who are quick to put out fires at work are great, but it's much better (and more productive) to work in a "fire-free" zone. The ultimate epiphany came to me when I read the following sentence in her article: "Ironically, the very skill of dealing with disasters masks the underlying need for it—the lack of planning and process—thereby discouraging management commitment to solving the real problem." Immediately I related and realized that problems could be avoided if I had better production processes. Teams that have processes in place to address problems before they arise are extremely productive and have high morale, which helps reserve adrenaline and patience for true emergencies.

If you seek out problems at work or are quick to reward triumphant workers after an all-nighter without questioning what led up to the overtime, it is possible that you're an adrenaline addict or an enabler of one. Take it from me, a recovering addict: Sometimes the stable predictability of a good work program is just as exciting as defeating problems.

Read more of Linda Hayes's article "Addicted to Adrenaline" at www.StickyMinds.com/11-2editorspick.



Francesca Matteu
Editor, StickyMinds.com
fmatteu@sqa.com

Quotables

"Your article reminded me of a question and its answer that was posed by one of my junior high school teachers:

Q: How many ways are there to look at an elephant?

A: 360 (one degree at a time).

I think focusing on the issue (the ant) and then taking a few steps back (the colony) is a good idea, and will help us discover better ways to test and find defects. Nice article!"

POSTED BY STICKYMINDS.COM MEMBER J. ALEXANDER ON

MICHAEL BOLTON'S ARTICLE, "LUCKY AND SMART."

www.stickyminds.com/11-2quotable1

"As usual, after reading one of your articles, I am left with the same questions when I am done. In no particular order they are:

* Why didn't I think of that?

* How did Johanna know I was having this problem? Is she watching me?

* Did I leave the iron plugged in?

Of course, I ask myself the last question even when I'm not reading your articles, so we will skip that one. The other two questions just show how you have your finger on the pulse of what a lot of us are going through and of how simply you can explain it. Thanks for the great read."

POSTED BY STICKYMINDS.COM MEMBER JIM LITTLE ON

JOHANNA ROTHMAN'S COLUMN, "ARE YOUR PANTS ON

FIRE, OR DO YOU SUFFER FROM SPLIT FOCUS?"

www.stickyminds.com/11-2quotable2

Books Guide

The STICKYMINDS.COM BOOKS GUIDE is one of the most popular areas on our Web site. With more than 800 books—including many that have been reviewed by thought leaders, industry experts, and your peers—the STICKYMINDS.COM BOOKS GUIDE is your first stop for finding a good read. Not sure what you're looking for? Browse books by topic, including:

- Project & Team Management
- Test & Evaluation
- Requirements
- Design & Architecture
- Development & Deployment
- Reviews
- Process Improvement
- Measurement & Reporting
- Security
- Defect Tracking
- Configuration Management

MARCH WEEKLY COLUMNS

In March, the first month of spring, we'll see new ideas sprout from requirements guru Ellen Gottesdiener, project and people management leaders Esther Derby and Naomi Karten, and testing talents Clarke Ching and Danny Faught. Every Monday new columns are posted on StickyMinds. Subscribe to the StickyMinds RSS feed to be notified when new columns and other articles are posted on the site.

www.stickyminds.com/rss/

BOOK REVIEW

About Face 3: The Essentials of Interaction Design

By: Alan Cooper, Robert Reimann, and David Cronin

Reviewed By: John VanNorman

This book should remain a classic for quite some time, as the subject of this book—interaction design—will continue to be relevant as long as people interact with computers. The book addresses the basic problem of “usability,” which is something I have been trying to come to grips with for many years. Trying to implement that concept is tougher still. The concepts addressed are pervasive in the industry. The book defines and addresses these concepts by providing methods of getting good user interface requirements and then turning those into good interface design.

The authors have a casual writing style that is easy to follow and understand. They completely describe interactive design concepts and methods. Basic human interaction concepts are addressed first, then the requirements process is discussed, and, finally, the design process is defined. Examples are used liberally to illustrate concepts. The authors' approach to these concepts is a little different from the norm. For instance, the book places a high value on developing characterizations about the target users. At first, I found this approach daunting, but I warmed up to the concept because, as developers, we rarely try to characterize the users of our systems. The authors showed the importance of doing that.

The book is intended for just about everybody interested in interaction design. There are no technical references, and the concepts are easy to understand. When the application of the concepts gets a bit complex, I think the authors do a good job of explaining how to execute the concept. The authors even write about design concepts for screen components such as controls and menus. This may be too much for the casual reader, yet designers, developers, and QA people in general should find this enlightening.

Using the knowledge gained from reading this book, I already have asked questions of my development staff that we've never asked before. So I intend to use this book as a “keep-at-work” reference guide and use it when wishing to delight our users with easy-to-use applications that can assist their business process.

Visit www.StickyMinds.com/11-2bookreview to post your comments on this book.



The list below features some of the more popular columns that StickyMinds.com members have been reading lately.

11 Sunny Skies or Storms? Project Weather Reports

By Johanna Rothman

www.stickyminds.com/11-2Top1

21 Reusability vs. Usability: Where to Draw the Line?

By Linda Hayes

www.stickyminds.com/11-2Top2

31 Scope Keep, Not Scope Creep

By Ellen Gottesdiener

www.stickyminds.com/11-2Top3

41 Manual vs. Automated Code Review: The Fight for Superiority

By Bryan Sullivan

www.stickyminds.com/11-2Top4

51 The Heart of the Argument: Getting Past Positions to the Sources of Disagreement

By Payson Hall

www.stickyminds.com/11-2Top5

Research Simplified

Check out the new white paper section on StickyMinds.com. Select White Papers from the Resources drop-down list to access the papers currently hosted on our site. Authors include top developers, project managers, and testers from companies such as Borland, HP Software, Rally Software, and Klockwork.
www.stickyminds.com/WhitePaper/



WEB SEMINARS

by StickyMinds.com & Better Software magazine

February 2009 StickyMinds.com &
Better Software magazine

Web Seminar Available On Demand

Watch the first Web seminar of 2009 all

over again ... or for the first time! "Overcoming Requirements-based Testing's Hidden Pitfalls," sponsored by MKS, Inc., is available on demand. In this Web seminar, you will learn how the thoroughness of test coverage is often compromised by requirements-based testing traps that are unapparent to testers, analysts, and managers. Robin Goldsmith, a respected authority on software testing practices and author of *Discovering REAL Business Requirements for Software Project Success*, will identify key sources of requirements-based testing oversights, including:

- Failing to distinguish real business requirements from product/system requirements
- Falling into the testability trap
- Equating requirements-based testing with a single technique
- Presuming one test per requirement or use case scenario suffices

Register and watch this presentation on demand: www.SQE.com/go?WS022409SM or check out other upcoming and on-demand Web seminars on StickyMinds.com at:

www.StickyMinds.com/Media/Webseminar.

Have you been wondering how you can get a daily dose of what's new and popular on StickyMinds.com and in *Better Software* magazine?

We've been reading your mind! StickyMinds.com and *Better Software* magazine are now on **Twitter**. If you're already on **Twitter**, follow @StickyMinds for regular updates about weekly columns, news, discussion boards, eNewsletters, and more, as well as information about *Better Software* magazine articles and Software Quality Engineering conferences. Even if you don't have a **Twitter** account, you can follow our **Twitter** feed at www.twitter.com/StickyMinds.



eNEWSLETTER EXTRA

A sampling of content from our eNewsletter archives

iterations: January 15, 2009 www.StickyMinds.com/enewsletter11-2

Agile in Motion: Music and Metaphor

Interview with David Hussman

David Hussman leads DevJam, a company comprised of agile collaborators working on assignments worldwide. In the following interview, he elaborates on Jonathan Kohl's "We Need More Metaphors" by answering our questions about connecting music to software development.

iterations: What are some things you picked up in the music industry that have transferred to your work in software development?

David Hussman: The recording studio is filled with a collection of bright, shiny, technical distractions. Producing records taught me that, while processes and technology were helpful tools, my greatest hits happened when I got to know the band, got to know their music, and went to see them play live, either in the rehearsal room or on stage. With this knowledge, I always did a better job guiding them through the recording journey, helping to select a set of techniques and gear, which helped bring the best they had to the final product.

I do the same thing in my coaching. My greatest coaching hits happen when I get to know the community before suggesting which practices or variations of agile processes (e.g., Scrum, XP) it should use. Even though I cannot see a project community perform, I can take a short trip to its world, so as to better understand who its members are, how they work, why they are moving toward agility, and where they see their strengths or challenges.

Instead of prescribing a path, I combine my experience with what I know of that community's world to suggest, teach, and coach a set of practices that help its members deliver more value in a more predictable fashion. Contrary to what the many agilists teach, this does not always mean faster. Similar to learning to play difficult music, many people need to slow down to deliver more; simply playing a song faster does not mean it is better.

iterations: Jonathan writes about the over-reliance on manufacturing as a metaphor in agile. How have you seen that metaphor used, both successfully and unsuccessfully?

David Hussman: I think there are many great tools we can draw from the manufacturing community. Trying to find and fix (or stop) wasteful work habits is a great one. I took this and many other gems from my first read of *The Toyota Production System* by Taiichi Ohno. When I engage with a community to help it improve, I do look for waste, but I also use Ohno's Five Whys to make sure I am not simply running toward the first solution from my past that might seem to apply for this community.

So, while I do see ideas from manufacturing helping tune processes, I also see them being applied in an ever-increasing and simplistic manner. We are not building cars! For instance, the manufacturing world has physical constraints that need not limit software development. Where automobile manufacturers may need to keep five radiators in place as a tool to make decisions at the last responsible moment, software developers can build systems in an evolutionary manner, making incremental changes to systems without the need to build five parallel systems.

I have seen software teams overly focused on process only, trying to fix problems by applying ideas from manufacturing (in the name of "optimization") that could have been fixed in a much simpler manner by employing a few development practices—many of which they simply did not know.

While there is evolution in manufacturing, once the product can be produced, a great deal goes into how to produce more units faster. For me, this is where a great deal of art can be pulled from software development when manufacturing ideas are simplistically applied. I have met many teams who have consistent velocity and who are iteratively and consistently producing more mediocre code iteration after iteration. In the name of being lean, they have leaned out all the incremental innovation that happens as you try to bring a product to life and keep it alive—instead of working hard to produce 10,000 units with the highest quality in the least amount of time.

iterations: Why is the concept of "metaphor" in general important to software production?

David Hussman: Software is an abstraction that cannot be simply defined with prose. Kent Beck tried hard to raise the value of metaphor. I think it failed because it was too difficult for many (most people do not consciously wield metaphor). If you take George Lakoff's approach in his book *Metaphors We Live By*, you find that metaphors are everywhere. In my coaching, listening for the active metaphors—those that communities live by—I find one more tool to help me make connections and improve my ability to coach more effectively.

Metaphor also helps feed creativity by allowing people to "think outside the box"—an excellent example of what Lakoff discusses in his book.

#1

Middleware

ORACLE® | bea®

- ✓ #1 in Application Servers
- ✓ #1 in Service Oriented Architecture
- ✓ #1 in Application Infrastructure Suites
- ✓ #1 in Enterprise Performance Management

ORACLE®

oracle.com/middleware
or call 1.800.ORACLE.1

The Ghost of a Codebase Past

by Pete Goodliffe

Nostalgia isn't what it used to be—and neither is your old code. Who knows what functional gremlins and typographical demons lurk in your ancient handiwork? You thought it was perfect when you wrote it, but cast a critical eye over your old code and you'll inevitably bring to light all manner of code horrors.

Programmers, as a breed, strive to move onward. Few software developers stick with the same codebase for a prolonged period of time. The average programmer tends not to maintain his own code for too long. Rather than roll around in his own filth, he moves on to new pastures and rolls around in *someone else's* filth. Nice. Of course, it's fun to complain about other people's poor code, as we easily forget how bad our own work was.

Revisiting your old code can be an enlightening experience. It's like visiting an old relative you don't see very often. You don't know him as well as you think. You've forgotten things about him like his funny quirks and irritating ways. And you're surprised at how he's changed since you last saw him.

Looking back at your old code might make you shudder for a number of reasons.

Presentation

This is only an issue for languages that sanction ASCII-based artistic interpretation. Indeed, I rather admire languages like Java and C# with a de facto standard presentation style. It avoids many of the fractures over coding styles found predominantly in the C and C++ camps.

```
class standard_style
{
    int variable_name;
    bool method_name();
};
```

Listing 1

```
class JavaStyle
{
    int variableName;
    bool methodName();
};
```

Listing 2

Some C++ programmers follow standard library layout like that in listing 1, and some have more Java-esque leanings like in listing 2. I know that my presentation style has changed wildly over the years, depending on the company I'm working for at the time. But as long as the style is employed consistently



ISTOCKPHOTO

in your codebase, this is a trivial concern and is nothing to be embarrassed about.

The State of the Art

Most languages have rapidly developing built-in libraries. Thirteen years of evolution took Java's class library from a few hundred helpful classes to an unfathomable plethora of classes. C# is a relative newcomer to the development world, but over several major revisions its library has burgeoned and the language has grown myriad new facilities.

Such evolution—especially rapid in a language's early development—can render your code anachronistic. Anyone maintaining it might presume that you didn't understand language or library features, when actually they did not exist when your code was written.

C# gained generics in version 2.0. The 2004 code you'd have written, like that in listing 3, would today be written as listing 4. Indeed, listing 3 is potentially buggy and distasteful. The state of the art advances much faster than your old code.

Idioms

You can't help it as technology changes under your feet, but it's perhaps most embarrassing to look back at old code and see how unidiomatic it is. Once you learn the correct idioms for the language you're using, old code can look quite wrong.


```
using System.Collections;

ArrayList list_of_colours = new ArrayList(); // note: untyped
list_of_colours.Add("Green");
list_of_colours.Add("Yellow");
list_of_colours.Add(Int(3)); // oops! Error
```

Listing 3

```
using System.Collections.Generic;

List<string> list_of_colours = new
List<string>();
list_of_colours.Add("Green");
list_of_colours.Add("Yellow");
list_of_colours.Add(Int(3)); // doesn't compile
```

Listing 4

Some time ago, I worked with a team of C programmers moving (actually, shuffling slowly) toward the brave new world of C++. One of the initial additions to their new codebase was called `max` and is shown in listing 5. (Do you know why we have all those parentheses? Post your answers in the comments section of my article on [StickyMinds.com](#).)

Much later, someone revisited that early code and, knowing more about C++, realized how distasteful it actually was. This person rewrote it in the more idiomatic C++ shown in listing 6. This actually fixed some very subtle lurking bugs, which are illustrated in listing 6.

```
#define max(a,b) ((a)>(b)) ? (a) : (b))

void example()
{
    int a = 3, b = 10;
    int c = max(a, b);
}
```

Listing 5

```
template <typename T>
inline max(const T &a, const T &b)
{
    // No brackets needed!
    return a > b ? a : b;
}

void better_example()
{
    int a = 3, b = 10;

    // this would have failed using the macro
    // because ++a would be evaluated twice
    int c = max(++a, b);
}
```

Listing 6

```
// We don't declare any max function

void even_better_example()
{
    int a = 3, b = 10;
    int c = std::max(a, b);
}
```

Listing 7

The original listing 5 version had yet another problem: The macro clobbered a name in the C++ standard library. This led to the even better solution shown in listing 7—Just use the built-in `std::max` function! It's obvious in hindsight—the kind of thing you'd cringe about now but had no idea about back in the day.

Design Decisions

Did I really write that code in Perl? Did I really use such an inefficient sorting algorithm? Did I really write that by hand, rather than using a built-in library function? Did I really craft that awful interface for the module?

This class of problem is typically very hard to repair without a major code rewrite. This highlights why it's so important to make good design decisions up front and to ensure that your code has good modularity and is easy to change.


Bugs

Perhaps this is what dragged you back to an old codebase. We've all been there: A new bug has been discovered that forces you to look at some archaic code. Coming back with fresh eyes uncovers obvious problems that you missed at the time. How could this old stuff ever have worked?

Looking back over your old code is like a time-lapse code review, but it's a valuable exercise. Perhaps you should take a quick tour through some of your old work. Do you like the way you used to program?

It's important to understand how times have changed, how the programming world has moved on, and how your own personal skills have improved over time. Perhaps you don't have the time or opportunity to revise old code now, but knowing where you've come from helps to shape where you're going in your coding career.

Like the Ghost of Christmas Past, there are interesting cautionary lessons to be learned from our old code—if you take the time to look at it. **{end}**



How does your old code shape up in the modern world? If it doesn't look too bad, does that mean that you haven't learned anything new recently?

Follow the link on the [StickyMinds.com](#) homepage to join the conversation.

Off the Trails

by Michael Bolton

It was a busy conference, with testers milling about the hotel hallways, moving from session to session. It was also a tester's dream come true, for there, at the top of the hallway, were two computers—Windows- and Internet Explorer-based information kiosks, each with its own screen, mouse, and full keyboard. This was like candy for a pair of enthusiastic testers like James Bach and me. We began by surveying the systems. Apparently we had very limited rights to do anything other than to operate the browser. Default Windows programs like Accessories were unavailable. Internet Explorer's File/Close menu option was grayed out. Control-Alt-Delete brought up a message saying that the Task Manager was disabled due to an administration policy. We observed that the service provider had prevented access to most of the Internet—the only sites accessible were those associated with the hotel itself, the chain to which it belonged, and some of the major airlines.

Conference participants had to walk past the kiosks to get to the presentation rooms. During the breaks, a small crowd would gather around the machines as we tested, and people would eagerly shout suggestions as to what to try next. It was fun, and the audience had some great ideas: "Where else can you go? Try clicking that link! Try right clicking! Have you tried to open a command window? What if you put CMD.EXE in the address line?" Most of our discoveries were about activities that we were prevented from doing, but we learned some other things, too. At one point, I noticed that the times on the two displays were different. "These browsers are running on different boxes," I said.

"Are you sure? How do you know?" James asked.

"If they were running on the same machine, they'd show the same time," I replied.

"True," James said. "Did you notice that until now we hadn't really consid-



ISTOCKPHOTO

ered the question of whether they were running on the same machine or different ones?" A good point; the box is especially black when you can't see the box at all.

I had to give a presentation, so I reluctantly left my self-appointed testing assignment. I returned an hour later, and as I approached the kiosks, James was grinning.

"Check this out," he said. "Put your hands on either side of the keyboard—nowhere in particular—and spread out your fingers. Now, just flap your hands lightly on the keys, as though you were playing the congas, but not hard." I did. To my astonishment, after a few drumbeats, the image on the screen seemed to rotate counterclockwise ninety degrees. The browser was suddenly sideways. Wha...? "That's how I found it. It took me a minute to figure out why it's happening," said James. "You try."

I looked at my hands. The left was over Shift, Alt, Ctrl, Caps Lock, Tab, A, and Z. The right covered Ctrl, Shift, Alt, the Windows key, and the cursor navigation keys. I tried pressing Tab, which

didn't seem to have an effect. The cursor keys—up, down, left, right? Nope. Ctrl with those keys? No. Alt with those keys? No. Combinations? Ctrl-Alt-Down? Yes. When I pressed Ctrl-Alt-Down, the display suddenly flipped upside down. Ctrl-Alt-Left rotated the image such that the top was on the left, Ctrl-Alt-Right put the top on the right, and Ctrl-Alt-Up set the screen right-side up. For a few moments we amused ourselves by showing the effect to the other testers. Then I began to wonder what might happen if we changed strategy and toured the keyboard systematically.

The Windows key brings up the Start menu, but in combination with other keys, it provides a little-known set of shortcuts. By default, Windows-A doesn't seem to do anything. Windows-B brings up the taskbar—think Windows-Bar. Windows-C doesn't do anything. Windows-D minimizes the applications on the screen to expose the Desktop. Windows-E brings up Explorer. Windows-F presents the Windows Find (Search) feature. At the kiosk, I walked through the alphabet from Windows-A,

but all of these combinations seemed to be disabled. I was losing enthusiasm for my systematic tour, until I got to Windows-L, for Login or Lock—and there, to my astonishment, was the Login screen.

At least it was passworded. I considered trying to guess the password—maybe it was some combination of “Admin” and the hotel name, maybe combined with a 1 or a 0. Going through all of the possibilities felt like too much work, and lunchtime was approaching—and besides, there was that button in the lower right corner of the screen that said “Turn off computer.” I considered the potential consequences for a moment, then I clicked the button. The machine rebooted. As it was starting up, I pressed the F8 key and got access to the command prompt. I now owned the machine. Time to stop.

I rebooted and left the system alone. To my relief, the Windows logo appeared, Internet Explorer came up, and the kiosk software started inside the browser. Once it did, we noted the support number for the company that created and maintained the kiosk. We had to believe that the service provider’s programmers and technicians didn’t know about this vulnerability or had forgotten to change the key combination to something more obscure. James called the company to alert it to the vulnerability. The first-level support technician didn’t understand the implications of what James was reporting, but her supervisor did and took the report seriously.

This is an example of what we can find with freestyle exploration. We had been working implicitly under a very open-ended mission: Learn something about this system starting from scratch. We designed and executed tests in parallel with interpretation of test results and with learning. We created questions and answered them on the fly. Sometimes—as in the case of discovering that the two systems were on different machines—we obtained the answer before we had framed the corresponding question.

Not knowing from the outset where we were headed, we used an approach that we’ve found effective for all forms of test design: *Start anywhere you like.*

Every journey begins with a single step. For the very first step of a freestyle exploratory session, we can choose something that might be interesting, fun, familiar, or useful; something that we might want to learn about; or something that we could infer might be important to some client. Start with a question, a conjecture, a specifically defined task, or a completely unmotivated activity and see where it takes us. Did we learn anything? Did we observe something surprising? Did we have a new idea? Did we suddenly recognize a new risk? Is there something here that’s worth recording for future reference or that might be a point of departure for a more specifically chartered session later on? What previous knowledge did we bring to the table? What do we know now that we didn’t know before? What skills might we want to practice? When we report our findings, how does our client react? If we wanted to do something like this next time, what would we do differently?

Whatever we discover, we may choose to focus on it, explore it, and learn something about it, or we may choose to refocus, step off in a different direction, and investigate something new. Note that we can follow this process with anything that we might wish to test or review—a document, a prototype, a flowchart, a whiteboard diagram, a new feature, a unit, a complete program, or a test idea itself.

Would we test an entire product using *only* freestyle exploration? Of course not. A good test strategy is driven primarily by risk. Many problems in software products are familiar, and we can be enormously productive by anticipating those problems and seeking them out,

aided by risk lists, coverage outlines, and other forms of checklists. Yet we must also foster the discovery of new risks, problems that we haven’t anticipated. We can’t ever be sure of what we’re going to find, and sometimes we can’t recognize what we’re looking for until we see it. There are at least two ways of addressing that problem. One is to remain alert, alternating between focusing and defocusing, switching between random and systematic actions as we follow our planned routes through the program. Another is to wander every now and then—quite deliberately—away from the plans that we’ve set for ourselves and see what we find hiding in the bushes when we’re off the trails. {end}



Whether testing a product for business or for pleasure, what problems did you find incidentally—or accidentally?

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

We deliver web applications

www.railsware.com/be-agile

railsware

Go, Team!

by Patrick Bailey

Robin finally sensed progress as she clicked OK to check in code. This was the last major defect on the list for the real-time inventory management system sold with the conveyor systems manufactured by Rolling Warehouses.

“Hey, champ!” Robin looked at the rear-view mirror clamped to her desk.

Buck Brinkmeyer was framed in the mirror with its prescient message: “Warning! Objects in mirror are closer than they appear!”

Buck was a 1962 all-American football player, Rolling Warehouses’s top salesman, perpetual coach-cheerleader, and grand critter of some animal lodge. His high cheeks lifted a smile that lit up like a thousand-watt light bulb. His square frame filled her cubicle’s entrance; Robin was trapped, no escape. Taking a deep breath, she swiveled around.

“Hi, Bucky!” Robin said, trying to sound welcoming, but the tension from previous confrontations was obvious.

Several months prior, Buck was out of control with upper management’s blessing. Sales were everything, and if Buck made a promise, everyone had to deliver on that promise. If a customer wanted a new software feature, it didn’t matter that Robin and her colleagues weren’t consulted first. They had to deliver, because a “promise from Buck is as good as delivered. Without sales, there is no company.”

“Just get it out the door” was heard too often. Robin had a passion for what software should be, but the company seemed to prefer self-inflicted disasters, and Buck’s sound bites exacerbated this. As working hours got longer and less productive, Buck cheered for the “All Stars on the dream team.” When shoddy work was accepted to meet a deadline, Buck would tell the company president, “A field goal still gets points on the board! We can get touchdowns only if we’re still on the field.”

The madness escalated when the IT department was reassigned to the chief

operating officer, well known for squeezing all that could be squeezed out of a factory. The COO’s regular exercises for efficiency gains even included smaller cubicles to save office space. When things were not getting completed on schedule, there were status meetings, which became more frequent with each promise Buck made to a customer. Just as Robin and two other developers submitted their resignations, the COO begged them to reconsider. He realized something wasn’t working and had just recruited Carolyn Hodges, a seasoned software development manager.

When Carolyn arrived, she knew results were “king” in this world. However, she immediately leveraged what the COO observed—there was absolutely no control in how software was developed, and it showed in the customer complaints. That much at least gave her the license to establish a forty-hour work week and iteration cycles. Carolyn ensured each cycle started with a level playing field for the business people and the developers to compare notes on what was most crucial and could actually be done. Once it became observable that each iteration was ending as planned (with expected “results”), Carolyn carefully facilitated adding more agile practices with each new success. Eight months after Carolyn’s arrival, Robin and the two other developers were still with Rolling Warehouses.

But now, here was Buck standing in Robin’s cubicle for the first time in months. “I’m not sure if I can visit right now,” Robin said.

“No problem. I’m going to huddle with Carolyn. Just wanted to say hi.” Buck gave his typical double thumbs up while mimicking a touchdown gesture.

Huddling with Carolyn? Now he’ll find out the opposing team has a coach, too!

Robin was concerned because she thought Buck had that look in his eyes. Is he testing the waters for another emer-



ISTOCKPHOTO

gency?! Given Carolyn’s assertiveness and relationship with senior managers, Robin was hoping this time he might feel the cleats running over *his* back!

Buck’s image returned to the mirror; this time Carolyn was standing with him. Robin spun around.

“Hey, champ! Thanks again for all your great work!” Buck said before he left for the factory production area.

Carolyn sat in the chair next to Robin’s desk.

“He seems so transparent,” Robin said.

“That’s just his personality. He really is a good salesman.” Carolyn said.

Robin rolled her eyes. “I could sell heaven, too, if someone else had to go through the hell of making it.”

Carolyn smiled sympathetically. “I guess there’s no easy way to tell you. Buck wants to make a promise to a customer.” Carolyn tossed Buck’s written proposal on Robin’s desk.

“You stopped him, right?”

“Not for now.”

“What! Are we back to jumping through hoops?”

“No, but follow me and bring that proposal. I’ll explain after a quick detour.” Carolyn led Robin to the factory’s production area. It was Robin’s first visit there since her initial orientation. There was Buck walking around—clearly in his element—giving high fives to line workers. To the workers, this recognition never got old. It was the highlight of their days.

“Does he have an off switch?” Robin quipped.

STORY LINES

- Through relationship building, a good manager encourages developers to understand their systems and the larger systems they live in.
- Learn from the past, but don't live in the past.
- Good leaders understand the forces for change, and that includes ensuring wins are visible through regular iterations of work.
- Remember a fundamental rule: Let the business people make the business decisions. Let the technical people make the technical decisions.

"Robin, take a good look at these people. Do you see that woman Buck is hugging? Her son just graduated from college. Showing up at 4:00 a.m. for the past twenty-four years and working any overtime possible made that happen. These people give to this place because

they have to, and they are thankful that Buck continues to make sales."

"Look, if you're guiltting me into working overtime ..."

"I'm not. I'll get to the point. We have a customer who will commit to a sale if we provide some new features, including software changes. Look at that proposal and determine what is doable and decide by tomorrow how much time you need to work through a few stories."

"By tomorrow? So we're back to ..."

"A big part of *when* is your decision. For now, let the others in our morning stand-up meeting know about this so they can give you input. We will *appropriately* work this into our planning cycle. Buck presented a good business case, but he has learned not to make promises without talking to us first."

Robin remained incredulous.

"Robin, agile is not about picking and choosing what we want to do. It's about creating relationships, doing the right thing the right way, *and* delivering. Buck and the senior managers now understand the type of work we do is different. They also trust us because the regular releases

are visible proof we know what we're talking about. They're even willing to concede that pair programming is not two people doing the work of one and that we don't need cubicles."

Robin's mouth dropped open. "No way."

"Yes, way. It was Buck who said it seemed to make sense. He feels better now about what he sells, too."

Buck noticed Robin and Carolyn across the factory floor and gave a broad wave of his hand. Robin glanced down at the proposal, looked back at Buck, and looked at the smiles on the faces of the people he was greeting. Robin held the proposal up and with her other hand gave Buck a thumbs up.

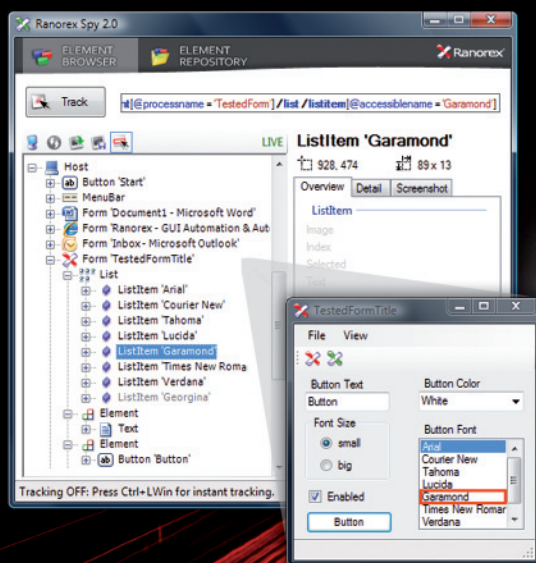
Touchdown. {end}

What was the defining moment that triggered a shift in your organization?

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

Ranorex 2.0

Next Generation Automated GUI Testing



- » Excellent Object Recognition
- » Object-based Capture/Replay Editor
- » Graphical Automation Editor
- » Professional Library for C#, VB.NET & Python



Download - Learn more
www.ranorex.com/trial



ECONOMY MEETS ECOLOGY.

Energy consumption in datacenters is expected to double in the next five years. Yet many businesses still don't know how much energy their IT is using. So how do you build and manage your IT to reduce energy consumption? With greener software from IBM: a complete range of energy-efficient software to optimize your infrastructure, boost business process efficiency and put practices in place for truly responsible collaboration. With energy at a premium, greener software can help shave millions off your IT and energy budgets. A greener world starts with greener business. Greener business starts with IBM.

SYSTEMS. SOFTWARE. SERVICES. FOR A GREENER WORLD.

Get our green strategy whitepaper at ibm.com/green/software







Lean Portfolio Management

Guiding IT Projects with Business Value

by Guy Beaver

Project Portfolios: Idea Inventories

An unintended outcome of project planning cycles is the deepening of the chasm between delivery organizations and their business clients. These planning efforts start with great intentions and tight development of business cases during the creation of project charters. Good communication typically occurs during project charter “season,” as business organizations lead value-driven discussions about capabilities and features needed to respond to market opportunities or competitor threats. This high-bandwidth communication often degrades as technical organizations require analysis time for each technical silo—data, QA, mid-tier, UI, and any other tier organization that exists. Once the analysis in silos begins, a clear, lean anti-pattern [1] appears and integration is delayed until some point far in the future of the project lifecycle.

Project-driven organizations typically focus on minimizing cost while maximizing utilization of resources, instead of focusing on speed and realizing the value of incremental delivery capabilities that are prioritized by return on investment (ROI). The faulty thinking is that careful, up-front planning can substantially minimize risk. In fact, delays caused by excessive up-front planning are often the biggest project risk encountered. This article dispels the myth of up-front planning as a panacea and offers an approach to driving the enterprise by prioritized business features that are managed in a visible portfolio. This lean portfolio management hastens the delivery of value to the business’s customers—both internal and external.

Project Portfolio Management

Organizations that embrace classic project portfolio management (PPM) must exhibit administrative excellence in order to be successful. The sheer bulk of projects that must be estimated, planned, and staffed creates a resource-loading problem that is difficult to solve and that is only as strong as its weakest link—the first project that gets behind. A common approach to justify PPM is to create the illusion of rapid delivery by releasing twelve- to eighteen-month-long proj-

ects every quarter. To the business, this makes a predictable pattern of change, which is desirable as long as high-business-value solutions are being delivered. But therein lies the problem; IT projects that began twelve to eighteen months ago were valued at market conditions that may have long since disappeared.

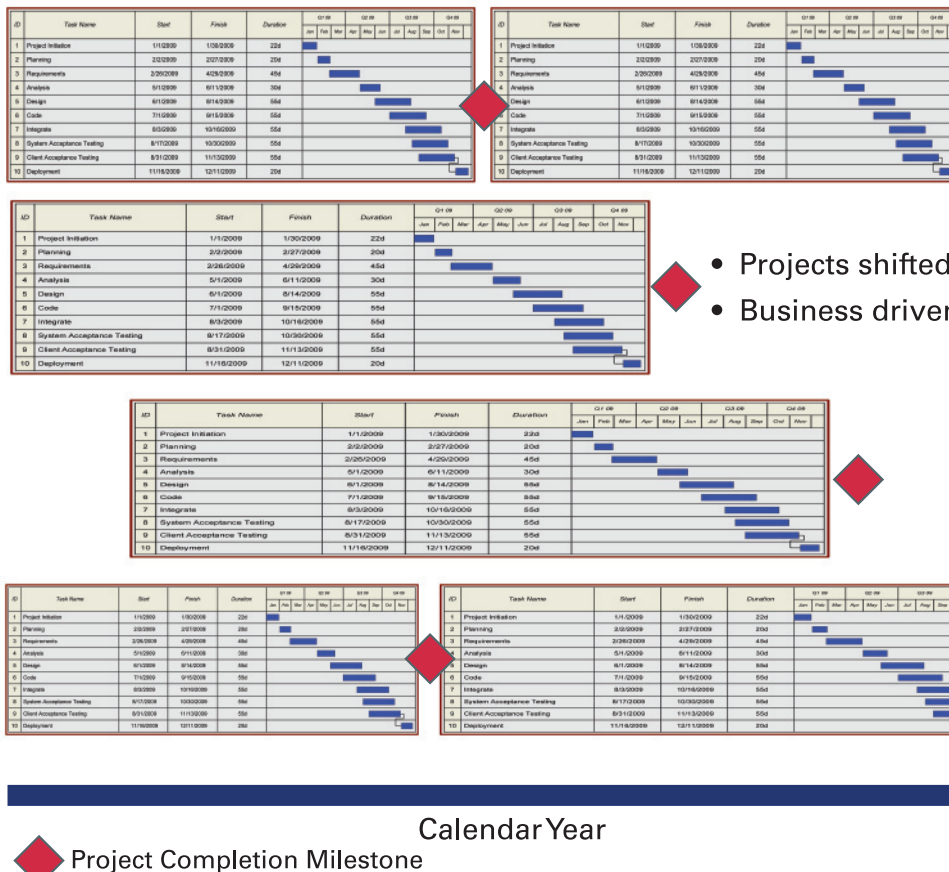
Another feature of the PPM cycle is that it creates a nearly unsolvable problem of a resource-loaded plan. Once projects are selected, an IT organization must load balance its staff by shifting the projects around so that resource demand is steady. This resource-driven behavior sacrifices business value in two ways: It batches together features into projects, holding high-value features hostage to lower-value, lower-return features; and heavy administration is required, which is manifest as a task-based approach.

Once the organization begins to focus on decomposing software development into sub-processes defined by these tasks, the opportunity for creating a stream of products is compromised. Projects get larger as the process—assigning and tracking tasks and delivering handoffs—slows down the enterprise. This slowing makes it necessary to create large batches of requirements, so the organizational skill for quick release of high-value features becomes compromised. This approach has inherent risks insofar as there are many dependencies between these projects—when one is delayed, a cascading effect of changes to the others occurs.

Mary and Tom Poppendieck [2] have applied lean principles to software development and point out that twelve to eighteen months to deliver an idea from “concept to cash” is wasteful. IT organizations that utilize PPM administer large batches of requirements, which are inventories of ideas. Lean tells us that creating and managing large inventories is wasteful. These large inventories of requirements are managed by big planning efforts that result in long delivery cycles. Discoveries are treated as errors in the process and are managed by restrictive change control.

Figure 1 illustrates the time dependencies of multiple projects managed by a yearly PPM cycle.

When viewed graphically, several of the challenges of this type of planning become apparent:



- Projects shifted to flatten resource load
- Business driven?

Figure 1: Multiple, parallel projects—timing and sequence of delivered projects “driven” by availability of resources, instead of business need

- The least valuable feature of a project holds all the other features “hostage” until it is completed, since the entire project is delayed until all features are completed.
- If one project is delayed, it can cascade into delaying other projects.
- It is difficult to insert newly discovered features that have higher business value to the organization.

The Lean Portfolio

In contrast to the PPM cycle, the *lean portfolio* approach allows stakeholders and clients to identify and prioritize features that create the highest ROI for the business. The lean organization is structured so that cross-functional teams can review and break down both business features and system dependencies in order to build minimal releasable software solutions. As opposed to PPM’s task-based administrative approach, the lean way is a results-based validation

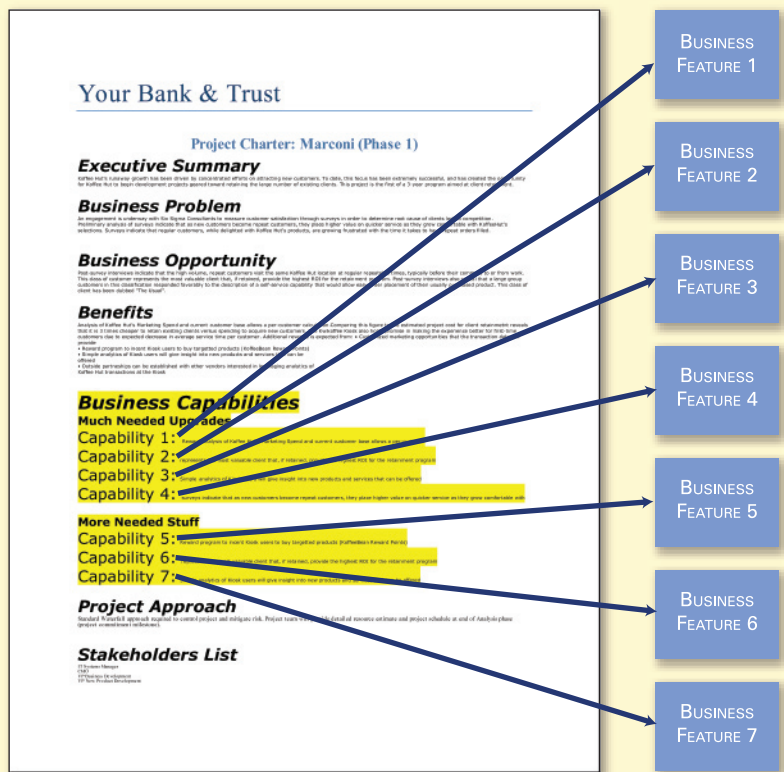


Figure 2: The project charter and how to transition to lean portfolio management

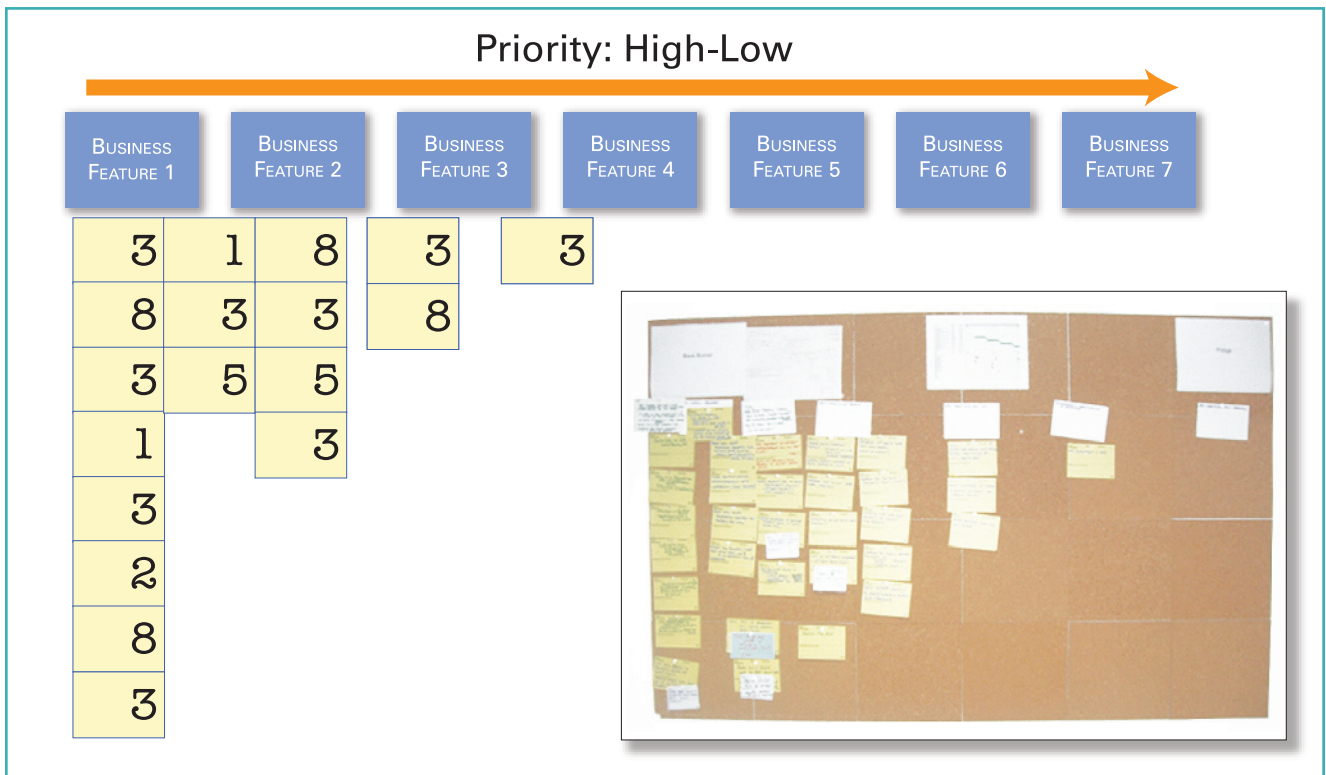


Figure 3: The lean portfolio—prioritized feature set, staged and decomposed into user and system stories (yellow cards)

approach. Status reviews are no longer based on tasks completed but instead on validation of technical results.

At first glance, the often-made assumption is that a premium must be paid—in terms of resources utilized and quality sacrificed—to deliver sooner. The promise of lean, seen as this article unfolds, is that you don't have to pay a premium but, in fact, just the opposite. By focusing on small, releasable features with the goal of getting software completed all the way through development, an agile team quickly exposes impediments to delivering rapidly. These impediments are normally hidden in large waterfall projects, as processes to transform handoffs create illusions of control.

A common misunderstanding in agile methods such as Scrum is that teams should not look ahead but instead should focus only on the work for the current iteration. This pattern is a leading cause of failed agile teams. Scrum doesn't say, "Don't do product planning or release planning." Similarly, lean tells us that it is important to look ahead, just not too far ahead. The lean product portfolio allows priorities to be set and elaboration of details to occur at the right respon-

sible moment. Agile methods allow a learning organization to emerge, which results in predictable estimation of features described at the capability level. These features can be decomposed in advance of the iteration in which they are actually implemented by establishing lean flow that is conceptualized by the planned release of features.

Let the Value Lead

Driving from business value means that IT is focused on rapid, high-quality delivery of the highest-value features. A clear violation of this pattern often occurs when a project is undertaken to rewrite an existing system using a new technology. Often, misguided intentions of strategic technology changes result in long, drawn-out projects that, in the end, deliver no new business value. Entire IT organizations can get drawn into these black hole projects that put business in the backseat with IT driving. The cost-minimization argument helps justify technology conversion projects, since no new requirements have to be gathered. Unfortunately, a very common outcome is that many of the unused product features—as well as system errors—are converted, also.

The right way to do conversion (or *any*) projects is to utilize and empower cross-functional teams to build incremental slices through the system so that the business stakeholders can validate realizable features as the technology is converted. This allows business value to drive and allows the development team to focus only on high-value, required features and not to waste time converting the unneeded features—or worse, bugs—of the system. It also allows the business to embrace market changes and opportunities that will arise during the conversion project. However, it requires an investment in training the technology organization in the use of agile design patterns and test-driven development. These agile engineering practices allow teams to create change-tolerant architectures that give them the confidence to make aggressive design changes, and suites of automated regression tests allow verification that nothing existing is broken as changes are implemented.

Organizational structure will indicate if IT is driving the business clients [3]. A clear indication of this is when an IT organization is structured around technical or functional areas, such as enterprise data, interfaces, or requirements.

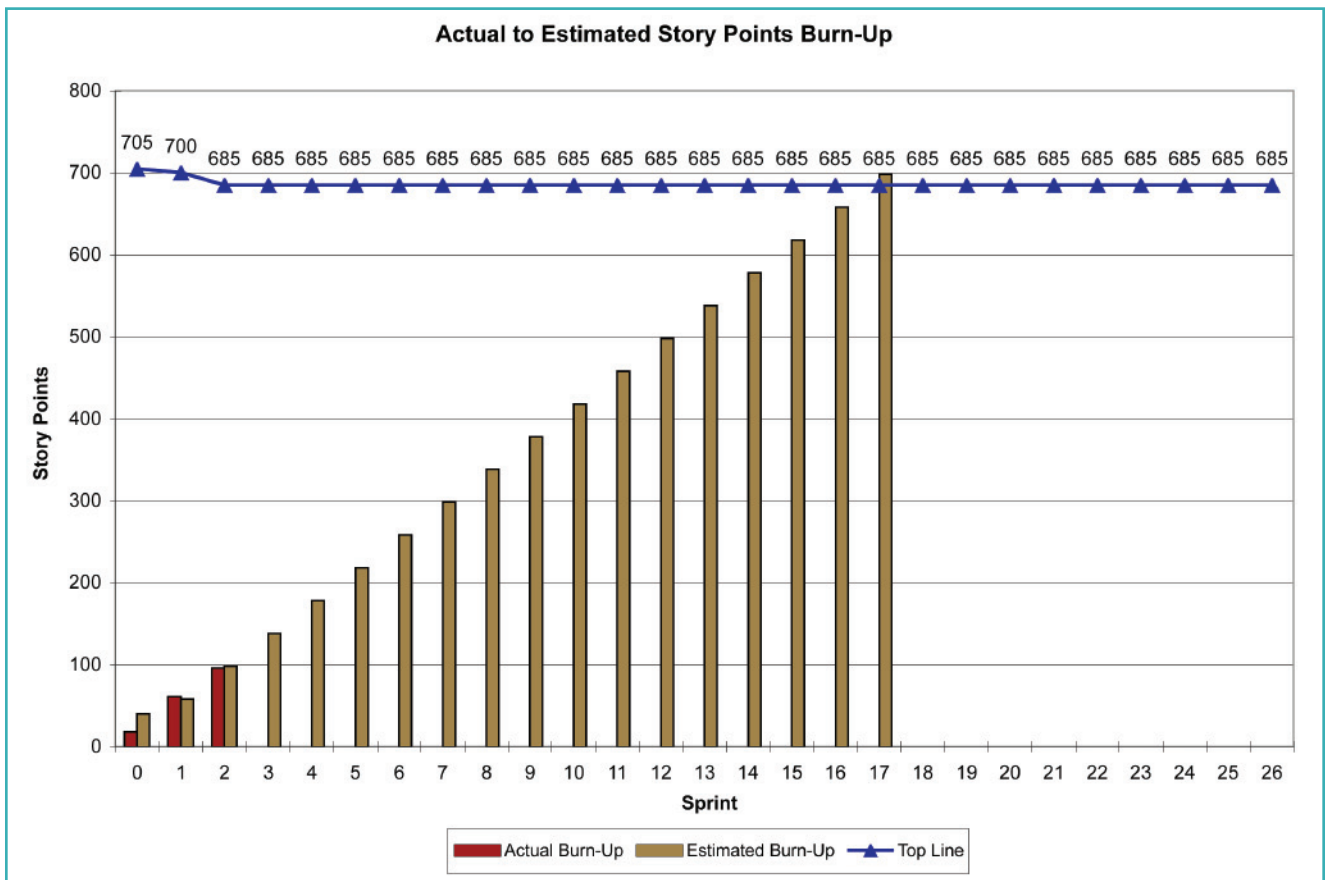


Figure 4: Each iteration delivers working software, measured in “story points.” The release status can easily be tracked in a burn-up chart.

These sub-organizations could appear in *any* IT organization, no matter what industry the business is in. These organizations fail a simple litmus test—clear line of sight with a business product. By organizing in this sub-optimal way, projects are delaying integration, which is a clear violation of the lean principle “optimize the whole.”

Another common pattern observed in PPM organizations is when large projects track status against a plan that indicates progress in each engineering phase. Agile principles emphasize that the most valuable indicator of status is working software. Large engineering efforts often show “green” status through the requirements, analysis, design, and build phases, but then mysteriously indicate “red” (as in two months behind) overnight. Lean claims that inventories hide true status and quality. Large inventories also mean delays in detecting errors, which cause waste. These two factors combine for yet another waste: large integration costs when unexpected events occur, due either to timing or to dependencies between the projects.

Lean Focus—Speed and Quality

An enterprise IT organization that is truly in sync with its business clients positions itself to help identify minimal releasable features and is structured to release them quickly. Organizations successfully making the transition to lean-agile discover that by going fast, impediments are quickly exposed so that the organization can self-correct. When the entire enterprise is focused on speed, market opportunities and threats can be leveraged and rapid savings or profits realized. In order to accommodate this, IT must begin to see delivery and quality as sustainable activities, which are constantly improved by short-cycle feedback loops. Once time to market becomes the focus, quality paradoxically goes up, as validation becomes more practical (through tight business coupling). Building little-used function also diminishes with this focus. This has the benefit of creating less complex systems because they are smaller, resulting again in higher quality.

A common pattern that emerges from organizations seeking this competitive advantage is that test-driven development and attention to agile design patterns are necessary for sustainable scaling of lean-agile teams. The reason is simple: Allowing productive teams to deliver quickly will not scale if they rapidly deliver disparate and redundant systems. For agile teams to scale successfully, they must pay attention to quality and integration. Quality comes from rolling out test-driven development approaches and leveraging emergent design patterns. This allows agile architectures that are geared toward change tolerance to emerge, giving business clients the competitive advantage of quick time to market.

The Lean Portfolio—Expected and Unexpected Benefits

The lean portfolio is a set or container of capabilities, defined at a high level, which the business requires to implement market or response strategies. These capabilities map loosely to capability state-

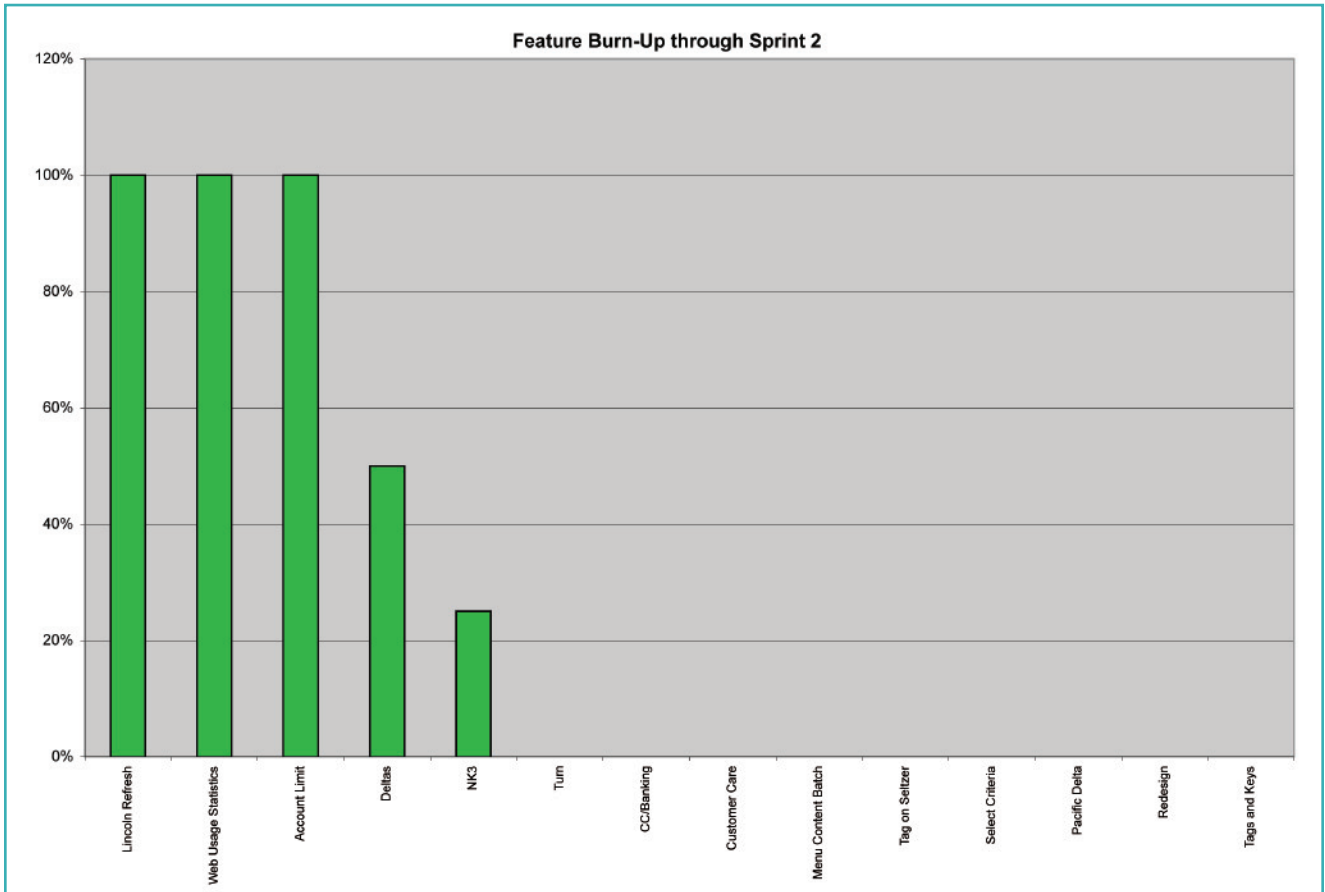


Figure 5: To ensure that features are being completed in incremental order, the burn-up chart should have an accompanying feature completion chart.

ments called out as business solutions in project charters or statements of work, as shown in figure 2. We call these capabilities “business features.” The lean portfolio gives the entire business and IT organization a focal point, where line of sight should be established for all work being undertaken, with the goal of minimizing work in progress and getting each capability completed as quickly as possible. The lean portfolio can be tracked for reporting, but the real value to the business organization is to create a visible display where business features can be listed to establish priority as well as technical effort required, as shown in figure 3.

By creating a lean portfolio of business features, several simplifying patterns emerge. Businesses can focus on prioritization and are given a clear picture of business value vs. cost. This is accomplished because agile teams learn how to provide estimates with just enough accuracy to determine the best value returned from effort. An effective practice for institutionalizing this estimation skill is through the use of story points [4].

The portfolio view also simplifies budget decisions, as funding centers can be realized by what percentage of features (along with cost) are staged in the portfolio. Once agile teams establish their velocity, which is effort or story points per iteration, accurate release plans can be created that give clear time to market for business goals, as shown in figures 4 and 5. If increased speed is required, then agile teams are scaled and integrated so that each team works toward visible release plans. This visibility is vital so that integration teams can look horizontally and leverage opportunities for refactoring and prevent duplicate work.

Dwight Eisenhower once said, “In preparing for battle, I have always found that plans are useless, but planning is indispensable.” So should an IT enterprise approach delivery of business solutions. A lean portfolio of features should be established to allow business and technology to explore ROI vs. technical risk. Correct selections of budgeted work can be allocated, as well as the creation of right-sized work that can be correctly estimated and pulled into a scaled agile

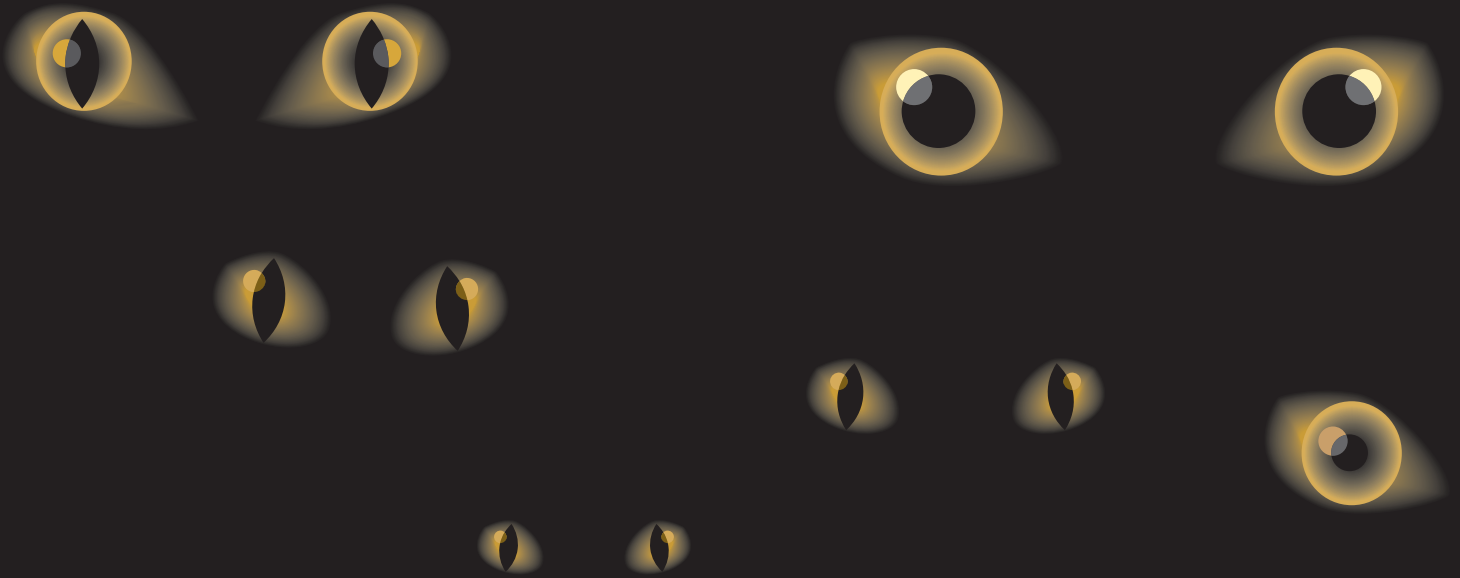
organization. A predictable release plan can be made visible allowing the enterprise to deliver technology solutions that are guided by business value. With a focus on correct engineering practices, enterprise agility will emerge, allowing the organization to be change enabled, which provides the competitive advantage afforded by quick time to market.

{end}

REFERENCES

- [1] Shalloway, Alan. “Lean Anti-patterns and What to Do About Them.” *Agile Journal*, August 8, 2007. www.agilejournal.com/content/view/553/76/
- [2] Poppendieck, Mary and Tom Poppendieck. *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley, 2006.
- [3] Beaver, Guy. “Knocking Down Silos: Transitioning the Enterprise to Agile.” *Agile Journal*, February 11, 2008. www.agilejournal.com/content/view/753/76/
- [4] Cohn, Mike. *Agile Estimating and Planning*. Prentice Hall, 2005.

TAMING THE HEADLESS



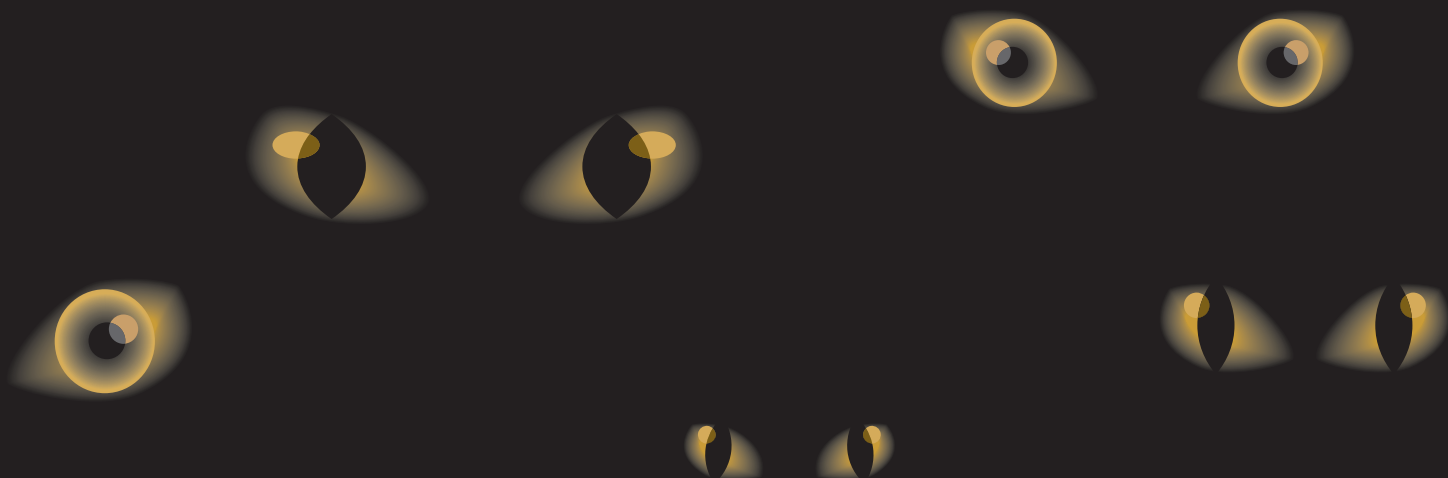
“When thundering hoofs echo through the darkness in the tiny valley of Sleepy Hollow, fearful farmers pull the covers over their heads. The only man foolish enough to be on the roads on such a night is Ichabod Crane. But is that ghastly figure bearing down on the teacher a wrathful spirit or just a jealous prankster?”

—The Legend of Sleepy Hollow

BEAST

A Proven Strategy for Testing Web Services

by David Fern



T*his passage* came to mind as I stood like Ichabod Crane while the project sponsor described an upcoming project that included Web services—a technology about which I had read so many mixed stories that I had no idea if it was the wave of the future or just another passing fad.

Time passed and my team successfully completed that first pilot project. We learned a lot along the way and were a bit more confident having one successful effort under our belts. New Web services projects kept coming, and with each new implementation we became more knowledgeable, adding new pro-

cesses and tools to our testing toolbox. By continually refining our processes, we now have a proven strategy for testing Web services that includes four distinct testing types, each building on the previous:

- Functionality testing
- Compliance and interoperability testing
- Security testing
- Performance testing

In this article, we will be working with a simple, fictitious Web service that implements the Sell Your Junk Web site. This Web service allows organizations and individuals to submit a picture (file)

of an item to sell, a text description of the item, and the selling price. The Sell Your Junk Web service forwards information to a billing Web service called BillEm, created by another group to charge the client a posting fee.

This Web service is very similar to those used by e-commerce sites that provide a way to post items for sale on their Web sites.

Looking a little deeper into the Sell Your Junk Web service, we see that it sends an XML request including the file name of the picture, item price, item description, and account number and receives an XML response. The details of the request are:

- The file name must be a zip file that is less than 255 alphanumeric characters long.
- The item price must be a number from 1 to 9999.
- The item description must be between one and 255 alphanumeric characters.
- The account number must be the submitter's account number and nine alphanumeric characters.

These criteria are combined into a Sell Your Junk SOAP request that looks like listing 1.

Functionality Testing

The theory behind testing Web services is not complex—send an XML-based request to a Web service, receive an XML-based response, and evaluate that response for correctness.

Web testers encounter challenges when testing the functionality of Web services because they are different from what Web testers are used to. Web services are headless—they do not have a Web user interface. They also are stateless—they do not remember information about past transactions. To mitigate these challenges, we need to test the Web

response, you can view the results in the test tool or use a file comparison tool to compare the response values to expected results.

HOW CAN WE TEST THE SELL YOUR JUNK WEB SERVICE?

First, create data tables like those shown in tables 1 and 2, as you would when testing any application. There are many ways to set up these data tables, but we have found that it works best to have columns corresponding directly to the XML request and response values. Using the Sell Your Junk example, there

are four values being sent in the XML request to the Web service (fileName, itemPrice, itemDescription, and accountNumber) and one value (statusCode), being sent back from the Web service in the XML response. This data table structure allows us to have the input data and expected values on the same line, making for easy execution and setup for automation in the future.

When you create the data tables, include valid and invalid data that test all fields as you would when testing any Web application, keeping in mind boundary values, special conditions, and project-specific requirements such as file types and sizes.

Now that we have data, we need a Web service test tool to input the data. The developers could create a GUI front end so you can enter data and check the responses, or you can use one of a number of Web service test tools.

Most Web service test tools do the following:

- 1) Read the Web services description language (WSDL)
- 2) Give the user a GUI to submit data
- 3) Create an XML file from the selections that the user has made
- 4) Send the XML request to the Web Service
- 5) Receive the XML response from the Web Service
- 6) Display the results

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:parameters.sellyourjunk.com"
xmlns:xm="http://www.w3.org/2005/05/xmlmime">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:SubmitItem>
      <urn:attachmentInfo>
        <urn:fileName>GadgetPhoto.zip</urn:fileName>
        <urn:itemPrice>1000</urn:itemPrice>
        <urn:itemDescription>Really cool gadget!</urn:itemDescription>
        <urn:accountNumber>ABC123DEF</urn:accountNumber>
      </urn:attachmentInfo>
      <urn:attachmentFile
xm:contentType="application/?">cid:123604199920</urn:attachmentFile>
    </urn:SubmitItem>
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 1

The Web service then returns an XML response to the original XML request. Here the Web service returns a status code of 0 for success and 1 for failure.

The Sell Your Junk SOAP response looks like listing 2.

By verifying the XML response you are able to determine that the application is functioning properly.

service as we would any Web application, verifying that all scenarios have been exercised and all transactions have been followed end to end. To perform functionality testing, create test cases and data for all valid and invalid test scenarios. Then, using a Web service test tool, send XML requests to the Web service and receive and verify that the XML response is correct. Once you have the

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ns1:ReceiptResponse xmlns:ns1="urn:sellyourjunk.com">
      <ns1:return>
        <ns1:statusCode>0</ns1:statusCode>
      </ns1:return>
    </ns1:ReceiptResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 2

Test Name	001-FV	002-FV
Description	This test validates that a 1 itemPrice will pass with a statusCode of 0. (Valid values for itemPrice are 1 to 9999.)	This test validates that a 9999 itemPrice will pass with a statusCode of 0. (Valid values for itemPrice are 1 to 9999.)
fileName	GadgetPhoto.zip	GadgetPhoto.zip
itemPrice	1	9999
itemDescription	Really neat gadget!	Really neat gadget!
accountNumber	ABC123DEF	ABC123DEF
statusCode	0	0

Table 1: Web service functional valid data table

Test Name	001-FI	002-FI	003-FI
Description	This test validates that a 0 itemPrice will fail with a statusCode of 1.	This test validates that a negative itemPrice will fail with a statusCode of 1.	This test validates that only a 4 digit itemPrice will fail with a statusCode of 1.
fileName	GadgetPhoto.zip	GadgetPhoto.zip	GadgetPhoto.zip
itemPrice	0000	-1000	99999
itemDescription	Really neat gadget!	Really neat gadget!	Really neat gadget!
accountNumber	ABC123DEF	ABC123DEF	ABC123DEF
statusCode	1	1	1

Table 2: Web service functional invalid data table

We will use a Web service test tool and point the tool to the WSDL URL or location, enter the security settings (discussed later), and be presented with an input screen that looks similar to the XML request. The screen may have an edit box or a “?” giving us a specific place to enter our data table values. Press the Run button and, like magic, the response will appear for you to verify. This is very similar to Web testing, but now we are using an “editable” GUI interface and, instead of being navigated to another page or checking a value on the screen, we need to verify the XML response.

Another tool that comes in handy for testing Web services is a file-comparison tool that allows you to compare the XML response to an expected XML file. We use the response fields in the data tables to verify field values, but we also want to verify that the complete XML schema is correct. This is very difficult

to do by eye. To verify an XML schema, save the expected XML response file and compare it to the actual XML response file that you receive. There are many free file-comparison tools, and most Web service test tools have file-comparison functionality built in.

Web service testing tools come in many different flavors, each with its own unique features. Some tools have features that assist in interacting with the Web services and include wizards that automatically create sample requests and GUI text boxes to enable you to enter required data easily into the XML request. Some tools create a “mock service” or stub allowing you to build functional tests before the actual Web service is ready.

Other tool features assist in building a test architecture or framework for data-driving tests, calculating WSDL and requirements coverage, and organizing test suites and test cases, as well

as complete test-execution reports. Most tools include functionality to test WSDL .NET and Java compliance, XML response verification, and XML schema verification.

If you are investigating Web service testing tools, you may want to consider the following sites that have Web services you can use to assess the capabilities and GUI presentation of different tools:

Xmethods

www.xmethods.net/ve2/Directory.po

WebServiceX.net

www.webservicex.net/WCF/default.aspx

These sites allow you to access a variety of different types of Web services that have been developed using different languages, methods, and toolkits. These provide you with a “testing ground” to investigate a tool’s look, feel, and functionality and a place to test out the tool’s features.

WS-I Profile Conformance Report

Report: WS-I Basic Profile Conformance Draft Report. This is a prerelease version and no statement can be made from this report on WS-I conformance

Timestamp: 2008-02-06T10:12:08-05:00

Copyright (c) 2002-2004 by The Web Services Interoperability Organization (WS-I) and Certain of its Members. All Rights Reserved.

Analyzer Tool Information

Version	1.1.0.22
Release Date	2005-02-23
Implementer Name	Web Services Interoperability Organization
Location	http://www.ws-i.org/deliverables/workinggroup.aspx?wg=testingtools

Analyzer Runtime Environment Information

Runtime Name	NET
Runtime Version	1.1.4322.2407
Operating System Name	Win32NT
Operating System Version	5.1.2600.0
XML Parser Name	NET
XML Parser Version	1.1.4322.2407

Analyzer Configuration Options

Verbosity: Trace

Figure 1: First page of a WS-I profile conformance report

Ideally, Web services testing should be performed directly, accessing its API, and not through a GUI that may filter or mask data. However, if you do create (or purchase) a tool that supplies a GUI, you now have the option of using automation test tools as you would with any other GUI application.

Compliance and Interoperability Testing

One of the promises of SOAP and Web services is interoperability, and this is key to the success of Web services.

After functionality testing, we know that the Sell Your Junk Web service works by itself, but now we need to verify that it works with the BillMe Web service developed by another group.

Web services rely on standards for SOAP, the WSDL, and security. See the StickyNotes for links to some of the common standards used in Web services.

Some of the interoperability challenges in testing Web services include understanding the impact of change and the increased number of communication paths caused by the increased number of APIs (one for each service). To mitigate the effect of these challenges, you must test the application and multiple Web services end to end, service by service, and interface by interface. Interoperability testing is performed by testing all valid and invalid multiservice scenarios,

as well as WSDL standards compliance. End-to-end, service-by-service, and interface-by-interface testing can be performed using the same tools used for functional testing; however, this time, the test connects Web services to other Web services completing the process. So we now may enter information into the Sell Your Junk Web service and check the output of the BillEm Web service, which is a database entry. This ensures complete, end-to-end testing.

One of the strengths of Web services is platform independence, meaning that the application can be .NET, Java, or any other language and still function properly. The way to verify this compatibility is by testing the WSDL to ensure that it is both .NET and Java compliant. WSDL validation is performed using a WSDL WS-I validation tool that is included as a part of most test tools. These tools analyze a WSDL to check that it conforms to WS-I standards.

Use the Interoperability Testing Tools 1.1 from the Web services Interoperability Organization Web site (www.ws-i.org), and run the test using the instructions provided with your tool. In most cases, you give the tool the location of the WSDL, the location of the interoperability testing tool file, and the type of assertions you want to show (e.g., all assertions, only passed, or only failed) and then execute.

Figure 1 shows the first page of a WS-I profile conformance report.

This report shows whether each assertion in the WSDL is compliant with WS-I standards. Repeat this test for both .NET and Java to ensure that the WSDL is WS-I compatible with both.

You are not required to test the WSDL for compatibility, and it does not need to be .NET and Java compatible to be a functioning Web service. However, if you are going to grow your architecture and move to SOA or hope to interface with other Web services in the future, it would be a best practice to start early with an agreed-upon architecture to take full advantage of the power of Web services.

Security Testing

Is the Web service safe? Can the Sell Your Junk Web service control who accesses it? The benefits of Web services also create the biggest security challenges:

- Loosely coupled services
- A collection of independent collaborating services
- Web services that may each require different authentication and enforce different security policies
- Web services that do not define how to do security, but rely on other mechanisms layered on top

As with SOAP and the WSDL, Web services have standards for security including the WS Security profile. A Web

Test Name	001-AUTH	002-AUTH
Description	This test verifies that valid user name being in group 1 can be properly authenticated.	This test verifies that valid user name being in group 2 can be properly authenticated.
User Name	Name1	Name2
Password	mypassword1	mypassword2
statusCode	0	0

Table 3: Web service authentication data table

service does not need to follow these standards to function; however, if you are planning for SOA or interoperability with other Web services, it is highly recommended. See the StickyNotes for links to a few of the Web service security standards.

Even though we discuss security, our focus is on authentication. Many Web services require a user name and password, allowing the service to control access. Another reason for authentication may be pay per service, where you pay for access to a Web service for a designated amount of time or certain number of uses with access controlled by a user account. In the Sell Your Junk Web service, the account number supplied will be charged a posting fee. However, there are many Web services that allow anyone to connect with no authentication check at all.

Many Web services use security or authentication tokens, which are embedded in the SOAP message header. These tokens may include information such as user name, password, or certificate allowing the authentication of the XML request. Since Web services are extensible, they are able to support multiple security token formats to accommodate a variety of authentication mechanisms.

A few security tokens used in Web services include:

- Username Token
- X509 Token
- Kerberos Token
- SAML Token
- SSL w/ 509 Client Authentication
- HTTP Basic Authentication

Mitigating the security risk can only be achieved through proper architectural

planning with centralized security management and then properly testing this architecture. Centralized security management becomes critically important as you move toward multiple Web services and SOA.

To validate authentication, we must understand the type of security token being used and what its property values should be. This may be new to many testers, since in Web applications you enter a user name and password and the application takes care of the rest behind the scenes.

The Sell Your Junk Web service uses x509 certificates with a user name and password, giving us two types of authentication tests to perform—test the certificate (which will be new to most Web testers) and test the user name and password or authentication testing (with which we are all familiar).

The x509 authentication testing can be easily performed using any commercial Web service test tool such as soapUI, Parasoft, Crosscheck, iTKO, or Mindreef. During validation, we want to test for a valid certificate, an invalid certificate, and no certificate at all. We will use valid functional test data and remove and replace the certificate through the mechanism the specific Web service testing tool provides.

User name and password testing is performed as you would test any Web login page. With the same tool used during functional testing, enter different user names and passwords and valid data, ensuring that the XML response is correct. A sample data table is shown in table 3.

Performance Testing

Will the Web service perform well

with a number of other Web services? The owner of the Sell Your Junk Web service needs to know that the service can support the load if a number of people submit items to the Web service at once.

Web services should be performance tested because of challenges inherent in the architecture of Web services. The loosely coupled, platform-independent functionality and connectivity is not free. It involves a substantial amount of communications overhead.

Major performance problems are often due to layer upon layer of services:

- Small services with large overhead
- Large services that are under-supported by hardware
- Services distributed across a network with their associated latency

Stress and performance testing are needed to find concurrency problems caused by the many different code paths and timing conditions.

A strategy for mitigating the risk of these challenges is to increase the intensity of tests including end to end, service by service, and interface by interface, ensuring all scenarios have been exercised.

Performance testers should feel right at home testing Web services, using many of the same tools and applying the same strategies—repetition, concurrency, magnitude, and random variation—used every day in their Web performance testing.

Repetition: Perform the same sequence of events over and over; for example, send the same XML request to a Web service again and again. This tells us if a service is fit for production, since typically Web services are used in a similar manner every time. We can repeatedly submit the same test that we created earlier during functionality testing.

Concurrency: Execute several different test scenarios simultaneously or simulate multiple clients performing different operations by sending multiple XML requests to a Web service at once. This tests multi-threaded behavior in which code in one thread may be interrupted by code from other threads. As testers, we should mix valid and invalid test data together.

Magnitude: Stress the system looking

for performance degradation by increasing the magnitude of the test. If a Web service processes a file, a magnitude test would test a huge file or large message size. Choose tests that can be measured and modified by a user, such as size of data, length of delay, amount of data transferred, speed of input, and variety of input. We can submit different sizes of graphic files and text strings.

Random Variation: Randomly mix the test sequence or combination of variables, allowing coverage of different code paths each time a test is run. Examples of using random variation include varying time between repetitions, number of repetitions, combination and number of Web services that are carried out together, or variable values.

Keep in mind that simple HTML/SOAP generators simulating many client connections generate high loads on the Web servers and may find problems with Web servers, but not with the Web services. So be sure you are testing what you think you are testing.

Taming the Web Services Beast

Don't let the headless "Web services beast" scare you as the horseman did Ichabod Crane. Web services are revolutionizing the way we do business and will be with us for a long time. You can tame this beast by creating a complete test strategy that includes testing functionality, compliance and interoperability, security, and performance using strategies similar to those used in Web testing. **{end}**

Sticky Notes

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

- Common standards
- Web services security standards

THINK YOU DON'T NEED TESTING TOOLS? WE BEG TO DIFFER!

Due to a series of unfortunate technical events, the contact email addresses and the trial/training information were listed incorrectly in the 2009 Tools Guide section of the January/February 2009 issue.

This version of the digital edition corrects these mistakes. We apologize for any inconvenience.
www.nxtbook.com/nxtbooks/sqe/bettersoftware0109

Don't drown in excessive software development and maintenance costs.

Grab onto the best life preserver in the software measurement & analysis industry.



Q/P Management Group, Inc.

Effectively implementing world class services and tools designed to manage risks, improve productivity, quality and to reduce costs.

Measurement & Estimation Programs

Software Benchmarking

Outsourcing Management

Function Point Analysis

Process & Quality Improvement

Training, Education & Mentoring

Trust Q/P Management Group, the Industry Leader, to keep you afloat.



Q/P MANAGEMENT GROUP, INC.



SEI Partner

For more information:

www.qpmg.com

moreinfo@qpmg.com

contact our office at 781.438.2692

AGILE SOFTWARE DEVELOPMENT TRAINING

Accelerate Your Career & Empower Your Team



BUILD-YOUR-OWN TRAINING WEEK

Click here for more information
on customizing your own
training week.

Maximize the impact of your training by combining
courses in one location to create a customized
training week. Pair two courses and save up to \$500.

For a complete list of courses available, visit
www.sqetraining.com or call 888.268.8770
or 904.278.0524 for pairing discount options.



www.sqetraining.com

Click here for a list of
all our courses.

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

AGILE SOFTWARE DEVELOPMENT TRAINING WEEK LOCATIONS

March 30-April 3, 2009

April 20-24, 2009

May 11-15, 2009

Washington, DC

San Francisco, CA

Chicago, IL

Choose from 4 specialized training courses:

THREE-DAY COURSES (Monday - Wednesday)

- Scrum Master Certification
- Design Patterns Explained


TWO-DAY COURSES (Thursday - Friday)

- Practical Test-Driven Development
- User Stories and Estimation in Agile Development

Click here for a list of all Agile courses.



Let SQE Training come to you. For more information about on-site training courses, contact
SQE Training at 904.278.0524 x212 or 233 or 888.268.8770 or email onsitetraining@sqe.com.



Building a Foundation for Structured Requirements

Aspect-oriented Requirements Engineering Explained
Part 2

By Yuri Chernak

Aspect-oriented requirements engineering (AORE) is a recently emerged and promising methodology that can help us improve requirements completeness, maintainability, and cost of development. Despite its benefits, today AORE remains virtually unknown to practitioners in the software industry. This two-part article is intended to fill this gap and explain the AORE concepts and techniques from the practitioner's perspective.

Introduction

In part 1 of this article (January/February 2009), I discuss that the IEEE Std 830 "Recommended Practice for Software Requirements Specifications" defines characteristics of good software requirements where two of those characteristics, completeness and maintainability, prove to be the most challenging to implement on software projects. These challenges stem from the fact that functional features of business applications commonly impact each other such that some features can be scattered across the application and tangled with other features. In the aspect-oriented methodology we call these scattered features *crosscutting concerns*. Such scattering and tangling effects present challenges

for developing complete and maintainable software requirements.

AORE specifically addresses these issues and provides effective analysis and specification techniques, shown in figure 1, that do not replace, but complement the existing methodologies. When following AORE in the analysis phase of the requirements process, we logically separate core features from crosscutting concerns. We then analyze the impact of crosscutting concerns on core features and document it in a specific form called a *requirements composition table* (RCT). Part 1 of this article discussed in detail the steps to create an RCT. An RCT presents a structure of a requirements model that has two dimensions: core features that are structured by application

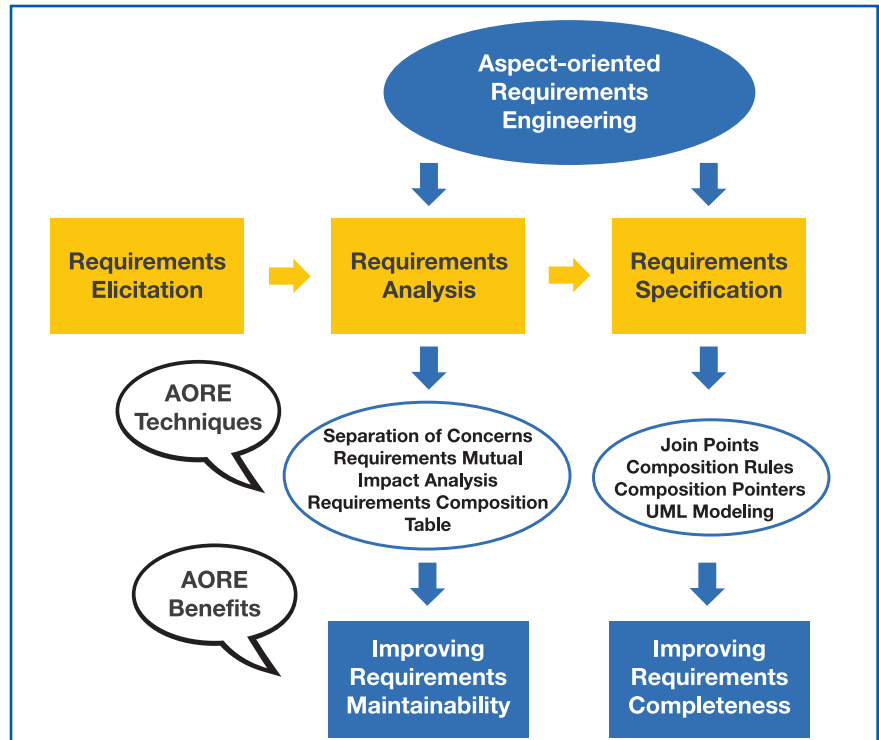


Figure 1: Summary of the AORE techniques

modules and supplementary features that are modularized into specific categories of crosscutting concerns. An RCT also shows for each core feature's context its composition with crosscutting concerns affecting a given core feature, thereby providing a structured and holistic view of the application functionality. Thus, as discussed in Part 1, an RCT is an important artifact of the analysis phase that can help us improve requirements maintainability.

Part 2 of the article focuses on the requirements specification techniques where AORE introduces the concepts of composition rules, join points, and composition pointers, as shown in figure 1. Below I explain these concepts and illustrate by examples how to apply them on software projects to produce more complete requirements.

Defining Composition Rules

After we have performed requirements analysis and produced an RCT, we already know which crosscutting concerns affect each of the core features. Now, in the specification phase we need to develop ideas about how crosscutting concerns affect core features and document their impact in software requirements specifications.

In general, the impact of crosscutting concerns on core features can be classified by the following three cases that I illustrate with examples from an imaginary hotel management system:

- **A crosscutting concern imposes a constraint on the entire context of a core feature.** For example, the crosscutting concern ET-Entitlements imposes a constraint on use cases “Check in Guest” and “Check out Guest.” This means that depending on the entitlement, some users can check in and check out a guest, while other users cannot. The feature “Check in Guest” can be executed only for the reservations with the status “Reserved” and cannot be executed for reservations with other statuses; correspondingly, the feature “Check out Guest” can be executed only for reservations with the status “In House” and cannot be executed for reservations with other statuses.

- **A crosscutting concern interrupts the flow of and changes the outcome of a core feature.** For example, the crosscutting concerns FV-Field Validation and CN-Connectivity are invoked when a user has completed a guest reservation and attempts to submit it. If any of these validations fails, it interrupts the flow and changes the outcome of the affected core feature.
- **A crosscutting concern adds detail to a core feature.** For example, sending reservation data to the central reservation system and guest rewards system is the outcome of a few core features such as creating a new reservation and checking in or checking out a guest. Details of these data exchanges between the systems are encapsulated and specified in the crosscutting concern SI-System Interface that adds detail and becomes a part of the affected core feature functionality.

Based on this classification, AORE defines three types of composition rules [1] and provides guidelines for applying them in core feature specifications regardless of what style we use—user stories, use cases, or traditional specifications:

- **Wrap:** This rule is used when a crosscutting concern imposes a constraint and controls the entire context of a core feature.
- **Override:** This rule is used when a crosscutting concern can interrupt the core feature flow and change its outcome.
- **Overlap:** This rule is used when a crosscutting concern adds detail to a core feature.

Example 1: User Story

To illustrate the use of composition rules, let's consider a user story that is a part of the imaginary hotel management system:

01.04 Check in Guest “A user can check in a guest who has a valid reservation”

Often on agile projects, such user stories are written by customers to capture their needs. A project team could then

use each story as the basis to plan the release scope, estimate the development effort, and implement and test this feature. However, this simple specification does not tell us the whole story. If we look at the same core feature in the RCT shown in figure 2, we can clearly see the other concerns such as entitlements, field validation, system interface, and so on that affect this user story and are part of its context.

To complete the user story details, we can discuss with customers the impact and realizations of applicable crosscutting concerns. First, we discuss how crosscutting concerns affect a given story, and we select composition rules based on the above guidelines. Second, we discuss and document crosscutting concern realizations to add details to the user story specification as follows:

- **ET-wrap:** To begin this story, the system validates the user privilege to check in a guest.
- **ST-wrap:** To begin this story, the system validates the reservation status that should be “Reserved.”
- **FV-override:** To complete this story, the system checks whether the reservation's data-entry fields have valid values.
- **DDV-override:** To complete this story, the system checks whether some fields have valid combinations (e.g., arrival date < departure date, departure date < credit card expiration date).
- **CN-override:** To complete this story, the system checks whether the front end stayed connected.
- **CC-override:** To complete this story, the system verifies that the selected room is not already taken by another front-desk clerk.
- **SI-overlap:** To complete this story, the system sends the reservation data to the central reservation system and guest rewards system.

Specifying realizations of crosscutting concerns means capturing high-level ideas about how these concerns affect a given story, whereas the complete details of these concerns can be encapsulated in their own specifications (e.g., supplementary requirements) or captured as tests when developers follow the test-

"01. Front Desk" Module												
List of Concerns	01.01. Create Guest Reservation	01.02. Update Guest Reservation	01.03. Cancel Guest Reservation	01.04. Check-In Guest	01.05. Check-Out Guest	01.06. Post Charges to Guest's Folio	01.07. View, Update Folio Charges	01.08. Create Message for Guest	01.09. View, Cancel Message	01.10. Add Travel Agency Commissions	01.11. View, Update Travel Agency Commissions	01.12. Manage Rooming List
Core Functionality	1	1	1	1	1	1	1	1	1	1	1	1
GUI Features	1	1	1	1	1	1	1	1	1	1	1	1
Crosscutting Concerns												
ET - Entitlements	1	1	1	1	1	1	1	1	1	1	1	1
ST - Status	0	1	1	1	1	1	1	1	1	1	1	1
FV - Field Validation	1	1	0	1	0	1	1	0	0	1	1	1
DDV - Data Dependency Validation	1	1	0	1	0	1	1	0	0	0	0	0
CC - Concurrency	0	1	1	1	1	0	1	0	0	0	1	1
CN - Connectivity	1	1	1	1	1	1	1	1	1	1	1	1
SI - System Interface	1	1	1	1	1	0	0	0	0	0	0	0

Descriptions of Crosscutting Concerns

ET—Entitlements: This concern relates to defining various user entitlements and specifying which core features can and cannot be executed by a given entitlement.

ST—Status: This concern specifies a reservation or other entities lifecycle that is commonly composed of various statuses that affect and constrain execution of core features.

FV—Field Validation: This concern relates to validating individual data entry fields.

DDV—Data Dependency Validation: This concern relates to validating a combination of related fields. For example, a reservation's check-in date should be before a check-out date, the check-out date should be before a credit card expiration date, etc.

CC—Concurrency: This concern relates to handling concurrent manipulation of the same data by multiple users.

CN—Connectivity: This concern relates to validating that the system's front end stays connected while the user completes a given transaction, and it defines the alternative behavior when the application goes into a disconnected state.

SI—System Interface: This concern relates to details of sending and receiving data to or from other systems (upstream or downstream). Such a concern can affect many core features that either use data from external systems or produce and send data to external systems.

Figure 2: RCT example for a hotel management system

driven development concept. In addition, the composition rules can help us better document how crosscutting concerns affect the story context. Once a user story is completed with the realizations of crosscutting concerns and their respective composition rules, we then can more clearly see the whole story. This, in turn, can help us improve effort estimation and quality of code, as well as develop more effective and complete tests.

Finally, to improve completeness of traditional requirements, we can use the same composition rules and follow the same guidelines to apply them as we discussed for user stories. In contrast, use case specifications present a more complicated and interesting case, which I discuss next.

Example 2: Use Case "U.C.01.04 Check in Guest"

To illustrate how composition rules can improve completeness of use case specifications, we will use as an example the same core feature that we discussed previously, although now its specification will be presented as the use case "UC.01.04 Check in Guest." The full text of this use case is shown in figure 3.

The issue with completeness of use case specifications is that use cases should focus on user/system interactions and should be documented at a particular level of abstraction. Thus, use cases are not intended to document all the details of an application's functionality related to a given use case. This means

that the context of use cases can be tangled with other concerns that are part of the affected use case functionality, but whose details do not belong to the use case specifications.

AORE resolves this issue by: a) using composition rules that reflect the impact of crosscutting concerns in use case scenarios and b) referencing realizations of crosscutting concerns that affect the use case context. As a result, we can produce a more complete specification that provides a holistic view of the application functionality in the use case context while keeping details of impacting crosscutting concerns in separate specifications. This technique can be performed by following four steps:

Use Case ID, Name:	UC.01.04 Check in Guest
Actor(s):	1. General Manager - Y 2. Front Desk Manager - Y 3. Front Desk User – Y (<i>primary actor</i>) 4. Housekeeping User – N 5. Support User – N
Pre-conditions:	1. User has a privilege to check in a guest (ET-wrap1). 2. Guest's reservation has a status "Reserved" (ST-wrap2). 3. Guest's name is displayed in "Today's Arrival" list.
Post-conditions:	1. Guest's reservation status is changed to "In House." 2. Guest's name is removed from "Today's Arrivals" list.
Normal Course of Events:	
<ol style="list-style-type: none"> 1. This use case starts when User selects Guest's name from the "Today's Arrival" list to access her reservation. 2. System displays Guest's reservation data and changes the reservation status from "Reserved" to "In House" (CN-override1). 3. User verifies/updates Guest's reservation data. 4. User selects a room for the guest. 5. User verifies/updates the guest's payment information. 6. User completes the check-in process and submits the reservation. 7. System validates the check-in information, sends the reservation data to other systems, and displays a check-in confirmation message (FV-override2, DDV-override3, CC-override4, CN-override5, SI-overlap1). 8. User acknowledges the check-in confirmation. 9. System brings User back to the main screen, removes Guest's name from "Today's Arrivals" list, and this use case ends. 	

Composition pointers as hyperlinks

Use Case Appendix. Realizations of Crosscutting Concerns

Composition Pointers	Realizations of Crosscutting Concerns	References to Supplementary Requirements
ET-wrap1	This use case can be executed only by the following user roles - General Manager, Front Desk Manager, Front Desk User. For other roles this feature is not available.	SR_ET: 01.05
ST-wrap2	This use case can be executed only when a reservation status is "Reserved." For other reservation statuses this feature is not available.	SR_ST: 01.01
CN-override1	System checks for the front-end connectivity before opening the guest's reservation.	SR_CN: 01.01
FV-override2	System validates individual fields—Guest Name, Guest Address, Number of Nights, etc.	SR_FV: 01.03, 01.09, 01.12
DDV-override3	System validates a combination of fields: Check-in Date < Check-out Date; Check-out Date < Credit Card Expiration Date;	SR_DDV: 01.04, 01.05
CC-override4	System validates concurrent selection of the same room for different guests.	SR_CC: 01.01
CN-override5	System checks for the front-end connectivity before saving a reservation.	SR_CN: 01.02
SI-overlap1	System sends the check-in information to the Central Reservation System and Guest Rewards System.	SR_SI: 01.02, 01.08

Composition pointers as bookmarks

Figure 3: Use Case "Check in Guest"

1. IDENTIFY JOIN POINTS

A *join point* could be either a use case precondition or a use case step that is impacted by one or more crosscutting concerns. It should not be confused with

the *extension point*, which has a conventional meaning in UML and is defined as a reference to one location within a use case at which behavior from other use cases may be inserted. Thus, a join

point indicates composition of a use case with the affected crosscutting concerns, whereas an extension point indicates composition of a base use case with either included or extending use cases.

To identify join points for a use case, we take each crosscutting concern listed in the RCT as applicable to that use case and, by going through the use case specification, we decide where a given crosscutting concern can affect the use case context. In our RCT example shown in figure 2, the first applicable crosscutting concern is ET-Entitlements. Now, we examine the use case specification shown in figure 3 and find that the first use case precondition relates to user privilege. Hence, the ET concern affects this precondition that becomes a join point. The next applicable concern is ST-Status, which also controls the entire use case context and affects the second precondition that becomes a join point as well.

One more example is the crosscutting concern FV-Field Validation. We find that step 7 in the use case scenario shown in figure 3 performs validation of the reservation's data. Hence, this step becomes another join point as it is impacted by the FV concern. We continue this analysis for all other applicable crosscutting concerns until we find all of their respective join points. It is possible that the same join point can be impacted by multiple crosscutting concerns (step 7 is an example). It is also possible that the same type of crosscutting concerns can have multiple join points in the same use case scenario (e.g., the crosscutting concern CN-Connectivity affects steps 2 and 7).

2. SELECT COMPOSITION RULES

Once we have identified a join point, we can analyze how the given crosscutting concern affects the use case and decide what composition rule should be applied. This decision is based on the composition rule guidelines discussed previously. Following these guidelines, we select the *wrap* composition rule for the ET and ST concerns as they impose constraints on the use case context. We select the *override* rule for the FV, DDV, CC, and CN concerns, because if any of these validations fails it interrupts the use case flow and changes the outcome. Correspondingly, we select the *overlap* rule for the SI concern, as it adds detail to the use case outcome related to the data exchange with the external systems. A complete view of the composition results can be presented as a UML diagram

Composition Modeling Using UML

Composition of crosscutting concerns with a given core feature, whether it is documented as a user story or use case, can be modeled with UML and using diagrams similar to conventional use case diagrams (see figure 4 below).

To produce such a diagram, we follow a simple convention:

- A crosscutting concern is shown as an ellipse containing the type of the crosscutting concern.
- Each composition is represented by using the relationship keyword (e.g., <<wrap>>, <<override>>, <<overlap>>) that reflects the composition rule type.
- Association notations:
 - When a crosscutting concern is composed with a core feature using the wrap rule, we represent it in the diagram as a dashed arrow pointing from the crosscutting concern to the affected core feature and indicating that the crosscutting concern imposes a constraint and has complete control over the core feature context.
 - When a crosscutting concern is composed with a core feature using the override or overlap rules, we represent it in the diagram as a dashed arrow pointing from the core feature to the crosscutting concern and indicating that the crosscutting concern is invoked in the context of the affected core feature.

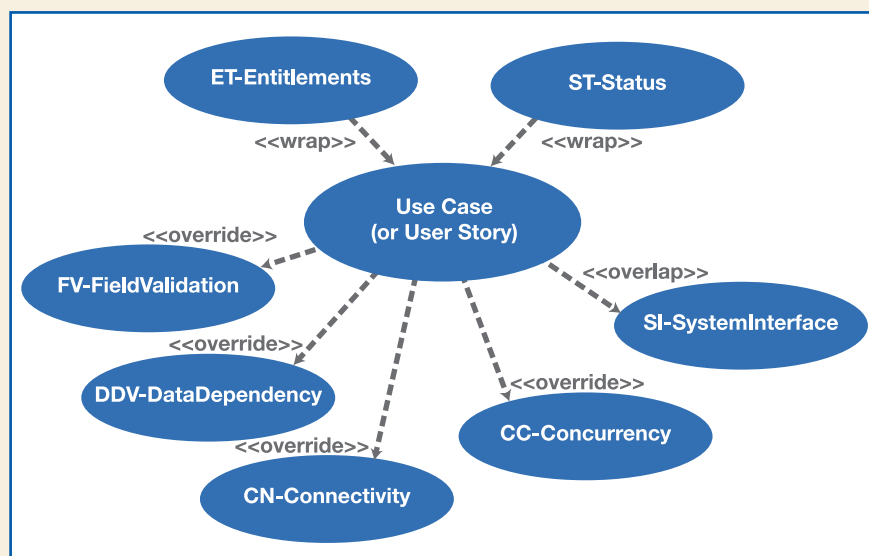


Figure 4: Composition modeling using UML

as shown in the side bar “Composition Modeling Using UML.”

When we have identified both a join point and a composition rule for a given crosscutting concern, we then document its impact by including a *composition pointer* in the join point. Each composition pointer is composed of the crosscutting concern type and its composition rule—for example, ET-wrap1 or FV-override2—as shown in figure 3. A composition pointer serves a dual purpose:

- It indicates which crosscutting concern impacts the given join point and how, so we can clearly

see what else we need to consider at this point in the use case scenario.

- It is used as a hyperlink to navigate quickly to details of a given crosscutting concern realization documented and bookmarked in a separate appendix of a use case specification, as shown in figure 3. This way, even if we include new steps or delete some steps in the use case scenario, the existing references (i.e., hyperlinks) to realizations of crosscutting concerns still remain valid.

3. DOCUMENT REALIZATIONS OF CROSSCUTTING CONCERNS

Up to this point, crosscutting concerns were just abstract categories of requirements that supported our analysis to answer such questions as: Which crosscutting concerns affect a given use case? and Where and how do they impact a use case context?

Now, the purpose of this step is to refine the impact of crosscutting concerns and specify their particular realizations in the use case context. As shown in figure 3, these details can be captured in a separate appendix section. This section has a table that, for each composition pointer presented as a bookmark, includes brief descriptions (just ideas) of particular realizations of crosscutting concerns at a given join point and references to detailed specifications of crosscutting concerns.

For example, the composition pointer FV-override2 that we included in step 7 in figure 3 points to the realization of the FV concern. Such a realization is specified in the appendix where the table provides detail about which fields should be individually validated at this join point. Another composition pointer DDV-override3 points to the realization of the DDV concern that provides details about which field combinations should be validated, and so on. Realization descriptions in the appendix do not need to be very detailed and can be captured at a high level of abstraction, which is sufficient for understanding the impact. For the complete details of crosscutting concerns, we refer in the appendix to their respective sections in supplementary requirements. However, such references can be completed only after the supplementary requirements have been developed.

4. ANALYZE AND RESOLVE CONFLICTS

As previously mentioned, a given join point can be affected by multiple crosscutting concerns. Step 7 in the use case specification is an example. In this case, we should analyze whether any of the crosscutting concerns can conflict with each other at the same join point. If so, we can specify the conflict resolution by listing composition pointers in a particular sequence, reflecting the order in which crosscutting concerns should

be invoked. This sequence can be determined based on the following guidelines:

- We place composition pointers for concerns with the *override* rule before the concerns with the *overlap* rule, because when a crosscutting concern overrides the outcome of a use case step, the overlapping crosscutting concerns will not be invoked, either.
- Composition pointers for crosscutting concerns having the same composition rule (i.e., override or overlap) are specified in the order of their business priority. In our example in figure 3, step 7 is a join point impacted by four crosscutting concerns with the same composition rule override. The sequence of their composition pointers at step 7 illustrates how the conflicts were resolved—i.e., the system invokes first the FV concern, then the DDV concern, and so on.

With this step, we have completed the analysis and specification of the use case “UC.01.04 Check in Guest.” By following the AORE techniques, we identified crosscutting concerns affecting the use case, selected related composition rules, modeled concern composition using UML, and specified the impact and realizations of crosscutting concerns by identifying join points and including composition pointers in the use case specification. This specification now captures the necessary details about the impact of crosscutting concerns, thus providing a structured and holistic view of the application functionality in the use case context.

In this article I discuss a new methodology known as aspect-oriented requirements engineering and use examples to illustrate its techniques and benefits. AORE focuses on resolving issues with the scattering and tangling of requirements to improve the modularization, maintainability, and completeness of requirements models. AORE does not replace but rather complements any of the existing requirements methodologies by adding effective analysis and specification techniques. As summarized in

figure 1 at the beginning of this article, when we follow AORE in the analysis phase, we separate concerns by core and crosscutting categories and then analyze and document the impact of crosscutting concerns on core features in the new analysis artifact called requirements composition table. In the specification phase, AORE introduces the concepts of join points, composition rules, and composition pointers that we use to document the impact of crosscutting concerns in the core feature specifications. Thus, by applying the AORE analysis and specification techniques we can improve the structure and completeness of requirements models that, in turn, can help us improve requirements maintainability and reduce the cost of developing requirements.

Finally, AORE is a relatively young methodology that is still evolving, exploring its benefits, and making its way to practitioners. At this point, AORE faces such challenges as consolidating various composition rules into a standard set of rules, adopting standard UML techniques to support aspect-oriented requirements modeling, and reconciling the differences between the two schools—i.e., two-dimensional vs. multi-dimensional separation of concerns to form a consistent methodology. Resolving these challenges is imperative for the faster adoption of the AORE methodology by practitioners in the software industry. **{end}**

REFERENCES

- [1] Brito, Isabel and Ana Moreira. “Towards a Composition Process for Aspect-Oriented Requirements.” Workshop at AOSD Conference, 2003.



SOFTWARE TESTING

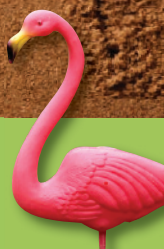
ANALYSIS & REVIEW

The Greatest Software Testing Conference on Earth

**STAR
EAST**



www.sqe.com/stareast
REGISTER EARLY AND SAVE UP TO \$300!



- 34 In-depth pre-conference tutorials on how to be more efficient and effective
- 40 Concurrent sessions on how to turn challenges into opportunities
- 5 Keynote presentations from top testing experts with experience in down times
- Networking opportunities to see how others are handling economic challenges
- Largest Testing EXPO event to help you find solutions
- Testing & Quality Leadership Summit on Friday to help you become a leader in the quality movement

99% OF 2008 ATTENDEES RECOMMEND STAREAST TO OTHERS IN THE INDUSTRY



New for 2009!
Testing & Quality Leadership Summit

BUILD YOUR CONFERENCE!

Conference schedule includes pre-conference tutorials, keynote presentations, concurrent sessions, summit sessions, and more!



MONDAY – TUESDAY

Choose from 34 half-and full-day tutorials that allow you to learn in-depth about specific topics.



Popular Tutorials Include:

Key Test Design Techniques
Becoming an Influential Test Team Leader
Reliable Test Effort Estimation
Measurement and Metrics for Test Managers
Risk-Based Testing: Focusing Your Scarce Resources

Adapting to Agile
How to Build, Support, and Add Value to Your Test Team
Essential Test Management and Planning
Just-In-Time Testing
Test Process Improvement
Fundamentals of Keyword-Driven Test Automation

WEDNESDAY – THURSDAY

5 Keynote Presentations
40 Concurrent Sessions
Networking EXPO
Special Events
...and More!



Covering topics like:

Test Management
Test Techniques
Agile Testing
Metrics
Personal Excellence
SOA Testing

Experience Reports
Static Testing
Performance Testing
Testing RIA/AJAX
Metrics
Outsourcing

Test Automation
Virtualization
Testing Web 2.0
...And Much More

THE TESTING EXPO

Visit Top Industry Providers Offering the Latest in Testing Solutions

TOOLS • SERVICES • TECHNIQUES • DEMOS

www.sqe.com/stareast



REGISTER EARLY AND SAVE UP TO \$300!

KEYNOTES BY INTERNATIONAL EXPERTS



The Testing Dashboard: Becoming an Information Provider

Randall Rice, Rice Consulting



Beyond Testing: Becoming Part of the Innovation Machine

Patrick Copeland, Google



What Haven't You Noticed Lately? Building Awareness in Testers

Michael Bolton, DevelopSense



Improve Your Testing through Automation

Jim Sartain, Adobe Systems, Inc.



Crossing the Chasm: Agile Transitions for Test Teams

Janet Gregory, DragonFire, Inc.

*"This conference was
EXCEPTIONAL! I will be back with
as many others as I can!"*

— Sam Johnson, QA Lead,
BioWare Corporation

FRIDAY

Testing & Quality Leadership Summit

New for 2009! Add a fifth full day to your conference event by attending the Testing & Quality Leadership Summit all-day Friday. Join senior leaders from the industry to gain new perspectives and share ideas on today's software testing issues.



Developing Jedi-level Test Talent: Practical Advice for Leaders

James Whittaker, Software Architect, Microsoft

Maximizing the Value of Testing to the Business

Marc René, Director, Billing Strategy, Business Architecture and Sourcing, MetLife Auto and Home

The ROI of Testing and Quality: A Business Executive's View

Jeffery Payne, CEO, Coveros, Inc.

Test Automation: Facts and Fiction for Executives and Managers

Michael D. Sowers, Sr. VP, Fidelity Investments

The Future of Testing—How Testing and Technology Will Change

Joachim Herschmann, Director of Product Management, Borland Software



WAYS TO SAVE ON YOUR CONFERENCE REGISTRATION



Bring Your Manager

Bring your manager and each of you saves an additional \$200. Any manager and practitioner registering at the same time save an additional \$200 off each registration.

Special Early Bird Offer!

Receive up to \$300 off your registration fee if payment is received on or before April 3, 2009.

PowerPass Discount

PowerPass holders receive an additional \$100 off their registration fee. Not a PowerPass member? Learn more about PowerPass at: www.stickyminds.com/powerpass.asp

Alumni Discount

STAR alumni receive up to an additional \$200 discount off their registration fee.

Certification Training + Conference + Summit

If you attend the "Software Tester Certification" Training Course AND the Conference, you save an additional \$300.

For **Group Discounts** or more details on our discount policy, please contact the Software Quality Engineering Client Support Group at sqeinfo@sqe.com or 888.268.8770 or 904.278.0524.



THE TESTING EXPO *May 6-7, 2009*

Visit Top Industry Providers Offering the Latest in Testing Solutions

Looking for answers? Take time to explore this one-of-a-kind EXPO, designed to bring you the latest solutions in testing technologies, software, and tools. To support your software testing efforts, participate in technical presentations and demonstrations conducted throughout the EXPO. Meet one-on-one with representatives from some of today's most progressive and innovative organizations.

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

Conference Sponsors:



Cognizant



Media Sponsors:

artima

BETTER
SOFTWARE
MAGAZINE

CoDe
www.code-magazine.com

InfoQ
Enterprise Software Development Community

Methods & Tools

StickyMinds.com

www.sqe.com/stareast



REGISTER EARLY AND SAVE UP TO \$300!

Agile Development Practices

BOULDER, CO—Rally Software has launched a Web site and online toolkit to calculate the cost-cutting benefits of agile practices. The new toolkit helps teams understand the financial impact of agile adoption and includes an ROI calculator, an industry study on the cost savings experienced by agile teams, and an Agile Success Guaranteed program.

Agile practices enable companies to build less but deliver the same value by focusing on the features that have the highest business value and not wasting money on the features that don't. Agile increases team productivity and product quality, which enables businesses to build more with fewer resources. A recent study conducted by QSM Associates showed that agile teams using Rally are 50 percent faster to market and 25 percent more productive as compared to plan-based or waterfall projects.

The online toolkit includes details about Rally's new Agile Success Guaranteed program, which eliminates the risk in adopting agile practices. The program enables teams to transition or scale their agile adoption with a three-month or six-month safety net, including: a thirty-day unconditional money back guarantee for product and services, early termination option within the first three or six months for a pro-rata refund, and a discount on product and service bundles. Rally is the only agile ALM provider to guarantee agile success, helping teams deliver their first agile projects in under sixty days.

The toolkit is available for free at www.rallydev.com/agilevalue.

TeamCity 4.0

PRAGUE, CZECH REPUBLIC—TeamCity is a continuous integration server and distributed build management tool available from JetBrains. Its key benefits include on-the-fly build feedback, faster builds and better scalability, clean code base, better control over large-scale environments, easy setup and adoption, and, finally, extensibility. Starting with v3.0, TeamCity has been free for small and medium-sized teams.

Features include:

- Build chains support, which allows the break down of a single

build procedure into several parts or builds that can be run on different build agents, in sequence or in parallel, using the same set of sources in all of them

- Tests re-ordering, which can determine a set of tests that is likely to fail and perform those tests first the next time the project builds
- Possibility to redo any build from a particular version control revision (history builds)
- Improved agents authentication mechanics, per-agent CPU-benchmarks, and agents overview statistics matrix
- Statistics for an entire project, and comparative statistics on a single chart
- New integrations, including FxCop and dedicated Rake runner, as well as improvements for old ones such as MSTest
- A new, highly improved Eclipse integration, which brings to the Eclipse world many cool features previously available only for IntelliJ IDEA users
- A lot of user interface improvements—new project page with project-centric information and statistics, dedicated page for viewing individual test statistics and details, and many improvements in dependencies management
- Some of the plugins bundled with TeamCity, such as integrations with Eclipse, ClearCase, and FxCop, are now open source
- Numerous other improvements, including notifications, VCS integration, a greatly improved search engine, and more

Visit www.jetbrains.com for additional information.

GUIDancer 3.0

GERMANY—Bredex GmbH announces the planned release date of GUIDancer 3.0, their automated GUI test tool, as the first quarter in 2009. New features in this release will include a new modeling perspective for model-based testing and a new observation mode to allow testers

to create reusable keywords from a running application. GUIDancer 3.0 offers comfortable, code-free, and agile testing to teams at all stages of the testing process.

Version 3.0 of GUIDancer will contain expansions in two directions. The first of these is model-based testing. Using the new modeling perspective, testers will be able to plan the structure of their tests and automatically transform the model into GUIDancer test cases. This new feature will extend the support that GUIDancer offers to teams testing early. Using requirements, activity, or UML diagrams available in the design phase, testers will be able to generate test cases to be executed when the application under test (AUT) is available.

The second new area supports testers later on in the test process—namely those starting testing when an application is already available.

Although no AUT is required to specify GUIDancer tests, this new observation mode will allow testers to create keywords using the running application. Keyword creation and refactoring is, of course, code free, which saves time and resources in testing and maintenance.

Visit www.bredexsw.com for more information.

CodePro AnalytiX

PORTLAND, OR—Instantiations, Inc. announces an upgrade to its comprehensive code quality product, CodePro AnalytiX. Through intense automation of audits, metrics, and best practices, CodePro AnalytiX ensures superior software quality and maximum developer productivity throughout the entire code development cycle. With this upgrade, Instantiations more than doubles the number of security audit rules to 150, bringing the industry's most comprehensive Java code auditing library to more than 1,100 rules.

Extensive security rules have been added in the categories of language semantics, Struts, and configuration files (Ant, Ivy, Maven, and WebSphere), while significantly expanding other categories including API usage, authentication, EJB security, file usage, tainted user input, and more.

Key enhancements include:

- Support for newly released IBM Rational Application Developer 7.5
- A new utility for converting supported FindBugs and PMD rule sets into CodePro AnalytiX audit rule sets
- Enhanced JUnit test generation to generate tests for EJB3 entity and session beans

Visit www.instantiations.com for more information.

SOA Test Version 5.5

NEWTON, MA—Version 5.5 of SOA Test from Parasoft Corp. includes closed-loop integration with runtime governance functionality from AmberPoint to Parasoft's service-oriented architecture (SOA) quality testing product. The merging of the two helps to create sophisticated testing scenarios for SOA applications running on complex distributed networks, so that system security is addressed.

The Parasoft-AmberPoint integration automatically emulates, or virtualizes,

services based on real-world historical data collected from the runtime environment. With this baseline established, teams can exercise distributed services in context without impact on the normal business transactions that continue. The runtime data is on individual services as well as end-to-end transactions. Other benefits are improved accuracy of the quality process, improved team collaboration, and reduced testing cycle and resolution times.

Visit www.parasoft.com for more information.

DevPartner Studio 9.0

LOS ANGELES, CA—Compuware Corporation announces the latest version of its award-winning code quality solution, Compuware DevPartner Studio 9.0. This new version improves an IT organization's ability to diagnose software security vulnerabilities, defects, and performance problems early in the development process—when problem resolution is most cost-effective.

DevPartner Studio 9.0 now supports 32-bit application development on Mi-

crosoft Windows x64 platforms as well as a number of new .NET Framework technologies, including:

- Visual Studio 2008 and Visual Studio Team System 2008
- Windows Server 2008
- .NET Framework 3.5
- Windows Presentation Foundation (WPF)
- Language Integrated Query (LINQ)
- ASP.NET AJAX Extensions

Visit www.compuware.com for additional information.

LDRA's New Automated Requirements Management

WIRRAL, UK—LDRA has created new technology that automates the administration and traceability of software requirements. This patent pending technology automates the techniques and tools for the integration of requirements traceability with software testing and verification. On completion, the technology will be integrated into TBreq, the portion of the LDRA tool suite devoted



Avoid Common Slip-Ups

Learn from top experts and practitioners with StickyMinds.com Web Seminars.



Our Web seminars feature experts who've created, tested, and refined solutions to common pitfalls in software development. Attend a Web seminar and you'll:

- Gain access to these solutions straight from your desk
- Interact through Q&A, Polls, and more
- Learn new tips and techniques to help solve your software development issues

Visit us today at www.StickyMinds.com/Webseminar

to requirements-driven testing.

LDRA's patent-pending automation of requirements-driven testing and traceability addresses an urgent need in safety- and mission-critical industries, such as avionics, medical, defense, and nuclear, where requirements traceability and verification consume a significant portion of project budgets. The technology automatically captures requirement information for software development projects from a wide variety of sources and simplifies the construction of a requirements traceability matrix.

The new technology enables developers to:

- Establish a baseline of selected software requirements from which development and testing will begin
- Delegate the tasks associated with each requirement to members of the development team
- Associate source code with each requirement, thereby automating the creation of trace relationships
- Refine source code mappings to

the level of class or even individual functions

- Verify requirements according to a wide variety of analyses and tests performed on the mapped code
- Automatically link requirements, test cases and code coverage results in a unified view of all levels of testing

TotalView Supports the HP Cluster Platform Workgroup System

NATICK, MA—TotalView Technologies has announced that its TotalView debugger now supports the HP Cluster Platform Workgroup System for midsize customers. TotalView provides users with the ability to debug parallel applications running on the HP system, which offers affordable, high-performance computing without the hassle, complexity, and expense of an enterprise-level system.

TotalView is a comprehensive source code analysis and memory error detection tool that can dramatically enhance

the productivity of small development teams. TotalView simplifies the process of debugging parallel, data-intensive, multi-process, multi-threaded, or network-distributed applications. Built to handle the complexities of the world's most demanding applications, TotalView offers a number of advanced features that help speed development and eliminate bugs quickly. It also provides visibility into thread creation and grouping via a full graphical user interface, giving developers the ability to analyze bugs to identify the root cause of problems and manipulate and control threads as needed.

Visit www.totalviewtech.com/products/totalview.html for more information.

Looking for an affordable, yet powerful solution to your automation challenges?

PHANTOM is the SOLUTION!

With today's economic challenges, you need to provide the best possible products on an ever tightening budget. Phantom provides powerful, affordable test automation to efficiently and reliably catch product defects before they reach your customers.

Visit www.phantomtest.com for a free trial download.



Powerful, reliable, flexible, and affordable. Phantom.

©2009 Phantom Automated Solutions, Inc.

10 Things You Might Not Know About

Agile Development

by Jeffery Payne

- 1** **AGILITY IS THE GOAL** The debate about what development practices are or are not agile is moot. Business stakeholders don't care what your development process is called; they just want a return on their software investment. Focus on becoming *more agile* instead of becoming "Agile" and your business value will improve.
- 2** **COLLOCATION OF TEAMS IS NOT NECESSARY** When building software, team rooms are definitely helpful for increasing agility but are not a showstopper. Distributed teams that focus on consistent, clear communication can use agile development practices to achieve business value.
- 3** **TEAMS ARE STILL ACCOUNTABLE** Implementing agile practices does not mean your software development teams are off the hook for on-time delivery. Agile greatly increases the visibility of day-to-day activities and drives teams toward higher productivity *and* accountability if implemented properly.
- 4** **BUSINESS SPONSORS NEEDN'T BE OVERBURDENED** Integrating business and domain knowledge into day-to-day project activities does not mean the process has to overburden your business sponsors. Business analysts can shoulder most of the day-to-day team interaction so business sponsors are called in only when really needed.
- 5** **UNIT TESTING IS NON-NEGOTIABLE** If you are not doing unit testing, you are not doing agile development. Unit testing is a cornerstone practice that makes other practices—such as continuous integration and refactoring—possible and valuable.
- 6** **AGILE SOFTWARE CAN BE SECURE** There is nothing about agile development that results in insecure software. Agile done right integrates secure development practices and security assurance into all development and testing activities.
- 7** **AGILE IS RIGOROUS** Don't be fooled into thinking that agile is a less-rigorous process than other development approaches. When done correctly, agile is a consistent, repeatable process with strong measurement of progress, risks, and productivity.
- 8** **GURUS DON'T HAVE ALL THE ANSWERS** Software succeeds or fails in the trenches where it is built. Many agile coaches haven't written a commercial software application in a decade. Trust and learn from those who *do* agile, not just talk about it.
- 9** **ARCHITECTURE AND DESIGN ARE STILL NECESSARY** You can't refactor your way into an initial architecture. Time must be spent during initial planning and iterations to get your architecture solid. From there, incremental development that includes refactoring can drive your software to completion.
- 10** **DOING DOCUMENTATION IS OK** Many software products must comply with standards, regulations, and corporate development policies. There is no reason that agile cannot be used to build these types of products as long as the appropriate documentation is created during the process.

Reloadable Test Data-O-Matic

by Tanya Dumaresq

Act now to take 50 percent off test execution time, virtually eliminate “works on my machine” arguments, and we’ll throw in load testing absolutely free!

Ladies and gentlemen, right now, you’re probably wondering if I’m crazy. “How can she make this incredible offer?”

Well, I’m here to tell you that with pre-created reloadable test data, it’s easy.

“What is this magical ‘reloadable test data?’” you ask.

Reloadable test data is test data that you reuse every time you test. Right now, many testers—and yes, even developers—tend to create test data on the fly each time they run a test.

Test Case ID 1 is an example test case based on an application that manages nurse/patient records in a health care system.

This requires a tester to create the following data (which would take at least ten minutes):

- Agency administrator
- Patient
- Two branches
- Two branch administrators with email addresses
- Nurses from patient’s old branch
- Multiple patient appointments with above nurses
- Various vital statistics for the patient

Chances are, this test would be run at least four times: a first pass, two regression or sanity tests, and one bug retest. Ten minutes times four test runs equals FORTY minutes.

But WAIT! With reloadable test data, creating that same data will take only TWENTY minutes. “How is that possible?” you ask.

By taking an extra ten minutes up front to set up and create this data in a reloadable format, you can reuse the test data four times—or even more.

Where does this extra time savings come from? I’m glad you asked ...

Take 50 Percent Off Test Execution Time

You, too, can create and use reloadable test data in five easy steps!

Step 1: As you write your test cases, simply keep a spreadsheet that records the entities you will need for each case and what special characteristics each entity must have. IMPORTANT! To maintain data freshness, do this as you write each test case—or else you will have to spend extra time trying to remember the needed data and you will end up with dry, cracked entities.

Table 1 shows part of a spreadsheet we’ve created for our application that

maintains nurse/patient records for a clinic.

But what does the “Is Changed” column mean? Can I just ignore it?

NO! That column is VERY IMPORTANT. Sometimes, in the course of running a test, the data for an entity is changed BY THE TEST. Perhaps Administrator 1 is deleted or Patient 3 is transferred to the care of Nurse 5. In a case where this happens, you will not be able to share this test data between different test cases.

Step 2: When all of your test cases are complete, it’s time to create the actual data in a format that is loadable for

Step	Action	Expected Results
1	Log in to application as Agency Administrator, and select a branch	Administrator is presented with a list of patients within the selected branch
2	Select one of the patients and transfer him to another branch	Verify that: <ul style="list-style-type: none"> - Appropriate branch administrators from new and old branches receive an email about the patient transfer - Patient can be found in the new branch but not in the old one - All of the patient’s appointments with nurses from the old branch are removed from the patient’s and nurses’ schedules - The patient’s vital statistics are transferred successfully

Test Case ID 1: Transfer of a patient to a new branch within an agency

Test Case ID	Data Type	Description	Is Changed?	Data Ref	Need to Know Data
1	Agency admin	Agency admin containing the two branches below	No	1	username, password
1	Branch admin	Admin of branch one containing patient below	No	2	name, email address
1	Branch admin	Admin of branch two that patient will be transferred to	No	3	name, email address
1	Patient	Patient with numerous nurse appointments and vital statistics, in branch one	Yes	4	name
1	Nurses	At least two nurses with various future appointments with above patient	Yes	5	names
2	6	...

Table 1

your application. For example, it could be another spreadsheet with sheets and columns that map directly to the tables in your application's database. As you create this data, keep track of its necessary identifiers by filling in the specifics in the "Need To Know" column from the original spreadsheet in table 1. BUT WAIT! Before you start, there's one more way to save time!

Look at each entity that you need to create—you could even re-sort your spreadsheet to help you do this—and compare them. Are any two exactly the same, right down to the "Is Changed" column (which must say "no")?

If so, you can create a single data entity for MORE THAN ONE TEST CASE. Imagine the savings! Instead of using ten minutes to save twenty, you can save sixty OR EVEN MORE!

Step 3. Go back to your test cases and insert the identifying information you will need when it comes time to execute the test case.

Step 4. Load the data into your application.

Step 5. Execute your test cases in re-

cord time! When the time comes to re-execute the same test, simply reload your original data and run it again.

Virtually Eliminates "Works on My Machine" Arguments

Hate getting your bugs returned to you marked "Not reproducible"? With reloadable test data, you may never see another bug like that. Here's how:

Once you've created your reloadable test data, don't keep it to yourself! We know you'll be tempted to keep this amazing wonder product a secret, especially from those pesky developers, but don't do it!

By sharing your reloadable test data with your development team, developers will see what you see when it comes to bugs. Using the same test data means getting the same results—virtually every time!

Plus, this means even more time savings for the team—if developers don't have to create their own unit test data, then they will have more time to fix the bugs you

report. Thanks, reloadable test data!

Act Now and Get Free Load Testing!

When testers create their data on the fly, they sometimes test only with very small pieces of isolated data—but with reloadable test data, you can test with a more realistic set of data, like your users will actually work with. And it doesn't require any extra time or effort on your part.

Reloadable test data also makes stress testing a breeze—no muss, no fuss, no icky clean up. Simply copy and paste the data in your spreadsheet as many times as you need to and then tweak the copied data to create new entities. At the touch of a button, "Nurse 1" becomes "Nurse 11."

Limited Time Offer ...

And did we mention that reloadable test data also slices, dices, and juliennes fries? To order, call the number on your screen. Hurry, before our stock runs out! {end}

Index to Advertisers

Better Software Conference & EXPO 2009	www.sqe.com/BetterSoftwareConf	9
Hewlett-Packard	www.hp.com/go/software	Back Cover
IBM	www.ibm.com/rational	24
Net Objectives	www.netobjectives.com	Inside Back Cover
Oracle	www.oracle.com/enterprise_manager/index.html	17
Perforce	www.perforce.com/perforce/schedule_demo.html	2
Phantom Automated Solutions	www.phantomtest.com	53
QP Management Group	www.qpmg.com	38
Railsware	www.railsware.com	21
Rally Software	www.rallydev.com/bsm	Inside Front Cover
Ranorex	www.ranorex.com	23
Scrum Alliance	www.scrumalliance.org	10
Seapine	www.seapine.com	1
SQE Testing Training	www.sqetraining.com/Testing	12
SQE Agile Training	www.sqetraining.com/Agile	39
STAREAST 2009	www.sqe.com/STAREAST	47-50
StickyMinds.com Web Seminars	www.stickyminds.com/media/Webseminar	52
TechExcel	www.techexcel.com	5

Display Advertising
advertisingsales@sqe.com

All Other Inquiries
info@bettersoftware.com

Better Software (USPS: 019-578, ISSN: 1532-3579) is published ten times per year. Subscription rate is US \$49 per year. A US \$35 shipping charge is incurred for all non-US addresses. Payments to Software Quality Engineering must be made in US funds drawn from a US bank. For more information, contact info@bettersoftware.com or call (800) 450-7854. Back issues may be purchased for \$15 per issue (plus shipping). Volume discounts available. Entire contents © 2009 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.



We are the only company that blends all of the leading **Lean-Agile** methods into a cohesive and comprehensive solution that is tailored to your organization.

Course Topics include:

Enterprise

Lean Software Development
Product Portfolio Management

Team

Agile Release Planning
Agile Analysis
Product Owner Certification
Scrum Master Certification

Individual

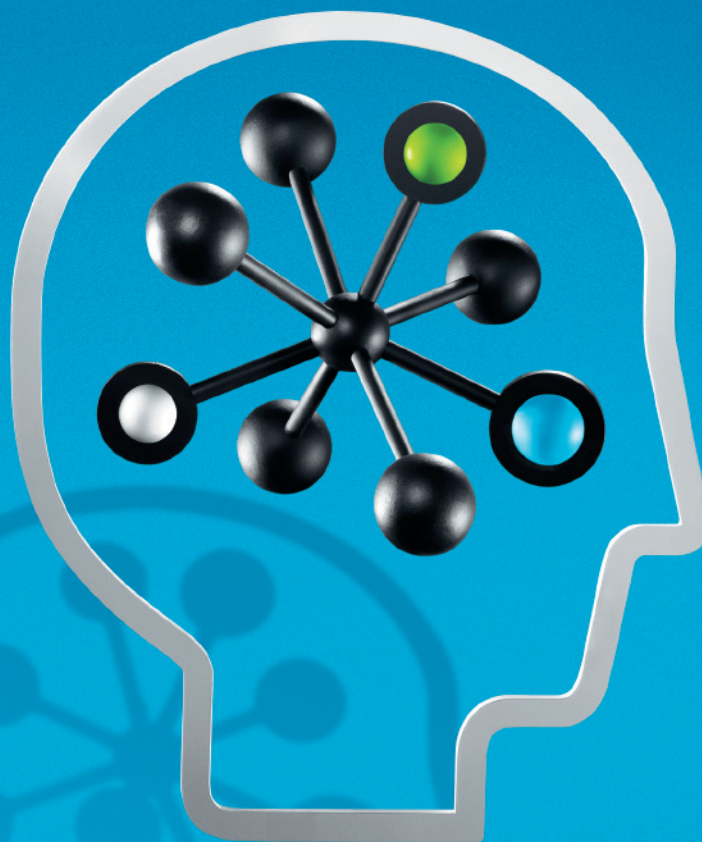
Test-Driven Development
Design Patterns

Online training
now available.

Save up to 80% on
training costs!

www.netobjectives.com/online

To set up a **free** one-hour assessment with one of our senior consultants, or for more information please contact Mike Shalloway at Mike.Shalloway@netobjectives.com or call 888-LEAN-244. www.netobjectives.com



ALTERNATIVE THINKING ABOUT APPLICATION LIFECYCLE MANAGEMENT:

Computers Don't Run Your Apps. People Do.

Alternative thinking is looking beyond the development cycle and focusing on customer satisfaction. Because the real application lifecycle involves real people – and the customer's perception is all that matters in the end.

HP helps you see the big picture and manage the application lifecycle. From the moment it starts – from a business goal, to requirements, to development and quality management – (and here is the difference) – all the way through to operations where the application touches your customers.

HP ALM offerings help you ensure that your applications not only function properly, but perform under heavy load and are secure from hackers. (Can't you just hear your customers cheer now?)

Technology for better business outcomes. hp.com/go/alm

