

# NOW I CAN PUT “NO-REPRO” TO BED.

## AUTOMATICALLY ATTACH CONTEXT TO BUGS.

Microsoft is making it easier to find and fix bugs. You now can add two new toolsets to help improve collaboration between testers and developers, reproduce bugs, automate repetitive tasks, and reduce the costs of setting up and maintaining multi-machine test environments.

**Microsoft® Visual Studio® Test Professional 2010** with simplified test planning, manual test execution, and rich, actionable bugs not only makes your job easier, but you can start to break the silos between test and development.

**Microsoft® Visual Studio® Lab Management 2010** helps teams rapidly provision virtual lab environments at a known state for test execution and build automation, effectively reducing wasted time and resources in your development and test life cycle.

**WHAT WILL YOU DO WITH VISUAL STUDIO 2010?**

Visit **[www.almcatalyst.com/test](http://www.almcatalyst.com/test)**

**Download Test Professional and Lab Management.**

Not sure how Visual Studio 2010 can help your team?

**Connect with an ALM expert**

**Sign up for an ALM workshop**

March/April 2010

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

# BETTER SOFTWARE

**MINIMIZE HANDOFFS**  
3 tips for agile teams

**GET IT TOGETHER**  
The power of collaboration

The Print Companion to

**StickyMinds.com**

## DEMYSTIFYING EXPLORATORY TESTING



**TAKE OUR  
CLOUD COMPUTING  
SURVEY!**



# NOW I CAN PUT “NO-REPRO” TO BED.

## AUTOMATICALLY ATTACH CONTEXT TO BUGS.

Microsoft is making it easier to find and fix bugs. You now can add two new toolsets to help improve collaboration between testers and developers, reproduce bugs, automate repetitive tasks, and reduce the costs of setting up and maintaining multi-machine test environments.

**Microsoft® Visual Studio® Test Professional 2010** with simplified test planning, manual test execution, and rich, actionable bugs not only makes your job easier, but you can start to break the silos between test and development.

**Microsoft® Visual Studio® Lab Management 2010** helps teams rapidly provision virtual lab environments at a known state for test execution and build automation, effectively reducing wasted time and resources in your development and test life cycle.

**WHAT WILL YOU DO WITH VISUAL STUDIO 2010?**

Visit [www.almcatalyst.com/test](http://www.almcatalyst.com/test)

**Download Test Professional and Lab Management.**

Not sure how Visual Studio 2010 can help your team?

**Connect with an ALM expert**

**Sign up for an ALM workshop**

# Save time while protecting software quality.



© 2009 Seapine Software, Inc. All rights reserved.

## Swiftcover.com cut their testing time in half with TestTrack Studio and QA Wizard Pro, while still providing the quality their customers expect.

Seapine's end-to-end Software Quality Assurance (SQA) solutions help you deliver quality products faster. Start with **QA Wizard Pro** for automated testing and add **TestTrack Studio** for issue tracking and test case management—integrated quality assurance solutions that together reduce testing time, saving you money and improving customer satisfaction.

- Reduce quality assurance costs with automated functional and regression testing.
- Manage test case development, defect assignments, and QA verification with one application.
- Track which test cases have been automated, schedule script runs, and link test results with defects.

“*So much of success boils down to time. QA Wizard Pro and TestTrack Studio allow us to be more profitable because we do more in less time.* — Test Manager, Swiftcover”

Learn how to test faster while protecting quality. Visit **www.seapine.com/betterswift**



# GIVE SPREADSHEETS THE BOOT!



## DevTest Studio

The #1 choice for test management and defect tracking.

### 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

Try DevTrack and DevTest live. Download free evaluation software. Watch a recorded overview demo.

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



# BETTER SOFTWARE

Volume 12, Issue 2 • March/April 2010



## CONTENTS



Take the  
*Better Software*  
magazine digital  
survey.  
This month's topic  
Cloud Computing.  
**pg. 23**

## in every issue

- Mark Your Calendar **4**
- Contributors **7**
- Editor's Note **8**
- Virtual Resource Shelf **20**
- Digital Survey Results **23**
- Product Announcements **39**
- FAQ **42**
- Ad Index **44**

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

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

## features

**26**

### COVER STORY

#### DEMYSTIFYING EXPLORATORY TESTING

Exploratory testing is a popular approach, but many testers secretly worry they might be doing it wrong. Jonathan Kohl addresses those concerns by explaining exploratory testing in ways that testers identify with.

*by Jonathan Kohl*

**32**

### AGILE TEAMWORK

Rather than rely on large handoffs between specialties, high-performing Scrum teams learn to do a little bit of everything all the time during a sprint. To do this effectively, teams must make three changes: shift from writing about requirements to talking about them, reduce the size of handoffs and make them more frequently, and pay more attention to the size of the product backlog items that they bring into their sprints.

*by Mike Cohn*

**36**

### WORKING TOGETHER—NOT JUST WORKING TOGETHER

People collaborate—and don't—in a variety of ways. Johanna Rothman examines what happens when collaboration isn't working, and how to make it work. Watch for several barriers to collaboration including those imposed on people by the organization itself.

*by Johanna Rothman*

## columns

**11**

### TECHNICALLY SPEAKING

#### A RELEASE WITHOUT A TESTER • *by Lee Copeland*

Inspired by the story depicted in a recent video store pick, Lee asks you to ponder this: "If you think testing is expensive, try the alternative."

**14**

### INSIDE ANALYSIS

#### SELLING TO YOUR BUYER • *by Scott Sehlhorst*

No matter how well you've built it, no users will benefit from your product unless you can convince the buyers to purchase it. Selling to buyers is different than satisfying users—and you have to do both well to succeed.

**17**

### MANAGEMENT CHRONICLES

#### HIDDEN MESSAGES • *by Dan Minkin*

Discover how an experienced test manager can gather useful information to guide decision making by looking at more than just the defect management system's data.

**43**

### CAREER DEVELOPMENT

#### THE UNSHREDDABLE RÉSUMÉ • *by Heather Shanholtzer*

The recent economic downturn has record numbers of job seekers pounding the pavement. Find out what you need to include on your résumé to increase your chances of getting out of the paper stack and into the building for that all-important interview.



## MARK YOUR CALENDAR



### software tester certification

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

### training weeks

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

#### Testing

**March 22–26, 2010**

San Diego, CA

**April 12–16, 2010**

Boston, MA

**May 17–21, 2010**

Chicago, IL

**March 22–24, 2010**

San Diego, CA

**April 12–14, 2010**

Boston, MA

**April 13–15, 2010**

San Francisco, CA

New York/New Jersey Area

**April 20–22, 2010**

Denver, CO

**April 25–27, 2010**

Orlando, FL

### conferences



#### STAREAST 2010

**Software Testing  
Analysis & Review**

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

**April 25–30, 2010**

Rosen Shingle Creek

Orlando, FL

#### Better Software Conference

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

**June 6–11, 2010**

Caesars Palace

Las Vegas, NV

#### Agile Development Practices|West

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

**June 6–11, 2010**

Caesars Palace

Las Vegas, NV

#### STARWEST 2010

**Software Testing  
Analysis & Review**

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

**September 26–October 1,  
2010**

Hilton San Diego Bayfront

San Diego, CA

#### Agile Development Practices|East

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

**November 14–19, 2010**

The Rosen Centre

Orlando, FL

## BETTER SOFTWARE

Publisher

**Wayne Middleton**

President

**Drew Thoenig**

Vice President of Publishing

**Holly N. Bourquin**

Editor in Chief

**Heather Shanholtzer**

#### Editorial

Managing Technical Editor

**Lee Copeland**

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**

Production Coordinator

**April Evans**

#### Circulation and Marketing

Circulation Coordinator

**Jamie Green-Gago**

Marketing Coordinator

**Stephanie Fender**

A PUBLICATION OF  
SOFTWARE QUALITY ENGINEERING



#### CONTACT US

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

Subscriber Services:

[info@bettersoftware.com](mailto: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



**Your global  
project team  
on the same page**

Agile Project Management  
Software from the Pioneers of Agile



**BOSTON**

Card #54  
"Under  
Development"



**BANGALORE**

Bugfix #31  
"In Testing"



**BEIJING**

Feature #76  
"done, Done,  
DONE"

Provide a shared  
workspace for Developers,  
QAs, BAs, Customers & PMs

Capture & Visualize  
all project activity

Manage XP, Scrum,  
Lean & Agile Hybrid  
projects

Get real-time intelligence  
with burn-down charts,  
velocity graphs, etc.

Leverage the industry's  
best User Interface

**DOWNLOAD FREE TRIAL AT**  
[www.thoughtworks-studios.com](http://www.thoughtworks-studios.com)



Copyright © 2009 ThoughtWorks, Inc. All rights reserved.



# Attend Live, Instructor-Led Classes Via Your Computer.

*New Live  
Virtual  
Courses Now  
Available!*



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

## Live Virtual Package Includes:

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



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



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



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



**JONATHAN KOHL** is a software testing consultant with Kohl Concepts Inc., based in Calgary, Alberta, Canada. Jonathan writes about and speaks on software testing. Read more of his work at [www.kohl.ca](http://www.kohl.ca). Contact Jonathan at [jonathan@kohl.ca](mailto:jonathan@kohl.ca). Jonathan would like to thank mobile tester Johan Hoberg and registered nurse Shelley De Boer for their review and contributions to this article.



An active professional in the computer field for forty years, **CLAIRE LOHR** has focused on software process improvement for the past twenty years. She is a Lloyd's Register trained ISO 9000 Lead Auditor and has been trained to perform software capability evaluations for the SW-CMM. Claire chaired the IEEE 829-2008 Working Group and has served on both the IEEE Computer Society's Software and Systems Engineering Standards Committee and the IEEE Computer Society's Standards Advisory Board. Claire is also an instructor with SQE Training.



Management consultant **JOHANNA ROTHMAN** is a regular StickyMinds.com and *Better Software* magazine columnist. She is the author of *Manage It! Your Guide to Modern Pragmatic Project Management*—winner of the 2008 Jolt Productivity Award—as well as coauthor of *Behind Closed Doors* and author of *Hiring the Best Knowledge Workers, Techies & Nerds*. Johanna is a host of the Amplifying Your Effectiveness Conference and has presented at numerous conferences. You can reach her at [jr@jrothman.com](mailto:jr@jrothman.com) or by visiting [www.jrothman.com](http://www.jrothman.com).



**DAN MINKIN** is a testing consultant with Certeco in the United Kingdom. As a test and project manager for the past ten years, his focus is on the practical and pragmatic application of testing and test management.



**SCOTT SEHLHORST** is an agile product manager, product owner, and business analyst and architect. He helps teams achieve software product success by helping them build “the right stuff” and “build the right stuff right.” Scott started Tyner Blain in 2005 to focus on helping companies translate strategy and market insights into great products and solutions. You can read more from Scott at [tynerblain.com/blog](http://tynerblain.com/blog).



**HEATHER SHANHOLTZER** is the editor in chief of *Better Software* magazine. You can contact her with questions, comments, and article ideas at [hshanholtzer@sqe.com](mailto:hshanholtzer@sqe.com).





### IN WITH THE NEW

In this issue, I am excited to unveil *Better Software* magazine's redesigned layout. It's the same high-quality content you've come to expect from us, with a couple of additions and a fresh look and feel.

Make sure to check out our new content including the Virtual Resource Shelf (p. 20), a section devoted to our authors' favorite resources. Be it a favorite book, blog, tool, or technique, now you'll know where the experts go to learn new skills.

Also new, is FAQ (p. 42), where we've asked one of SQE Training's highly experienced instructors to answer the question he or she is most frequently asked during training sessions. Chances are it's a question our readers have been dying to ask, as well.

Beginning this issue, we've added a survey exclusive to the digital edition. Download the digital edition to answer the survey question, and then check out the next issue—print and digital editions—to see the results.

We are always looking for feedback from our readers in order to provide you with the content you need to build better software. Please feel free to email me questions you would like answered in any of our new sections. Be sure to specify in the subject line: Virtual Resource Shelf, FAQ, or Digital Survey.

As always, I hope you enjoy this issue of *Better Software* magazine. I hope you find this issue's articles to be informative and useful in your quest for quality. Email me to let me know how you've put this issue to work for you.

Happy Reading!

A handwritten signature in dark ink that reads "Heather Shanholtzer". The signature is fluid and cursive, with the first name being more prominent.

Heather Shanholtzer

HShanholtzer@sqe.com

# Automate Your UI Testing with Ranorex



## Object-based Capture & Replay Editor

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



## Automated Testing of Web & Windows Applications

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



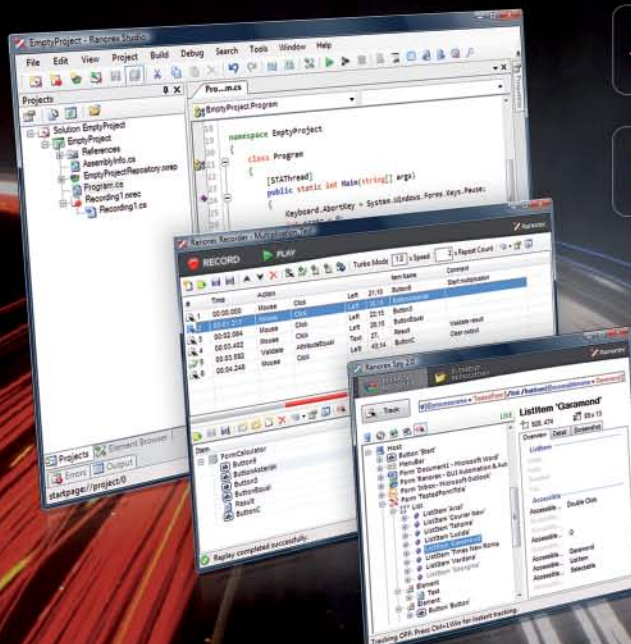
## Maintainable UI Object Repositories

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



## Write tests in C#, VB.NET and IronPython

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



Get your 30-day Trial  
[www.ranorex.com](http://www.ranorex.com)



Watch Demo Video  
[www.ranorex.com/demo](http://www.ranorex.com/demo)

VISIT US!





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

## Definitions

! Standard Defines

```
Include "RESPONSE_CODES.INC" Include "GLOBAL_VARIABLES.INC"  
CHARACTER*512 USER_AGENT Integer USE_PAGE_TIMERS CHARACTER*  
CHARACTER*1024 cookie_2_0 CHARACTER*1024 cookie_2_1 Timer T_
```

## Code

```
!Read in the default browser user agent field  
Entry[USER_AGENT,USE_PAGE_TIMER Start Timer T_OBFUSCATED  
PRIMARY GET URI "http://yahoo.cHTTP/1.1" ON 1 &  
HEADER DEFAULT_HEADERS &  
,WITH {"Accept: image/gif, image/xbitmap, image/jpeg, image/p  
"application/x-shockwave-flash, application/msword, /*/*", &
```

## or this:

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



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

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

# A Release Without a Tester

Imagine a world with no testing, no reviews, no inspections.

Now, sit back and watch what happens ...

by Lee Copeland | [lcopeland@sqe.com](mailto:lcopeland@sqe.com)

Recently, I drove down to the video rental store. All my favorite movies—*Plan 9 from Outer Space*, *Glen or Glenda*, *Santa Claus Conquers the Martians*, *Troll 2*, and *Escape from L.A.*—were checked out, but I found a movie I hadn't seen before—*A Day Without a Mexican*—and thought I'd give it a look.

The premise is simple: One day, without warning, everyone of Latino descent in California, a third of the state's population, mysteriously disappears. As a pink fog surrounds California cutting off both travel and Internet access, cooks, gardeners, nannies, farm and construction workers, entertainers, athletes, as well as the fastest growing market of consumers, vanish.

California is in shock. Experts pose questions and offer answers: A mass UFO abduction? Genetically targeted bio-terrorism? The Rapture and Latinos are the chosen ones? Or perhaps they just left because they were tired of being underpaid and taken for granted.

The film is not very good (although it did win the award for best editing at the Guadalajara International Film Festival). It can't decide whether it wants to be a goofy comedy, biting satire, or thoughtful social commentary. Ella Taylor of the *L.A. Weekly* described the movie as "A terrific premise, mangled to a pulp, then beaten to death in this forced mockumentary." *E! Online* was less kind, stating, "Day not only lacks Mexicans, but also good acting, sharp storytelling, and humor." Marjorie Baumgarten of the *Austin Chronicle* wrote, "Its narrative concept will entertain for a while, but eventually you will long to disappear with the rest of the Mexicans."

And now for something completely different. (Don't worry; it's like a Tom Clancy novel. This all comes together in the end.)

"Testing costs too much." "Testing takes too long." "Testing is a giant sinkhole for our organization's resources." Heard these expressions of frustration before? Sure. If you're in the testing profession, you've heard them repeatedly. And, if you're like most testers, you've quickly retorted, "Does not!" or some other equally well-reasoned, quantitatively based re-

sponse. Our difficulty is that most organizations have never attempted to quantify the value of testing. The cost—all those people and tools—is simple to compute. But the value—the economic benefit that comes from testing, minus its cost—is more difficult.

Every time testing discovers a defect that is repaired before it escapes to the customer, we have saved money for our organization and, therefore, created value. Specifically, we have saved the cost of dealing with the defect report, re-creating the problem, tracing from the failure to the underlying de-

fect, fixing the defect, testing the fix, testing the test that tests the fix, and re-distributing the software. Quantifying the value of testing is simply a matter of identifying, collecting, and summarizing those costs. A simple clerical task. Well, actually, a complex clerical task. It seems like a lot of work, so most organizations don't do it. I suggest we don't attempt to do it.

Rather, I propose a grand experiment. Instead of doing the work to quantify the value of testing, just ship the next release of your product with no testing at all—a release without a tester. (I told you it would come together.) No reviews, no inspections, no unit tests, no integration tests, no system tests, no acceptance tests, no performance tests, no security tests, no

usability tests, no tests of any kind. Then, sit back and watch what happens.

I predict a major catastrophe—systems that give incorrect results while appearing to run normally, systems that freeze and give no results at all, massive loss or corruption of data, and correct results presented in indecipherable, confusing, and unusable ways. The aftermath will be the disintegration of organizations, the collapse of economies, and the loss of lives. It will be a sight to behold—even better than the movie—a truly convincing display of the value of testing.

Of course, if this is too radical an approach, you can do that boring old "bean-counting" exercise of value calculation. But I keep hoping someone will try my experiment. **{end}**

**"I propose a grand experiment. Instead of doing the work to quantify the value of testing, just ship the next release of your product with no testing at all—a release without a tester."**

# BETTER SOFTWARE CONFERENCE

Conference Sponsor:



June 6-11, 2010  
Las Vegas, NV



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

REGISTER BY APRIL 9, 2010 AND  
**SAVE UP TO \$400**  
GROUPS OF 2 SAVE EVEN MORE!

## KEYNOTES

BY INTERNATIONAL EXPERTS



**Tim Lister**  
*Atlantic Systems Guild*



**Payson Hall**  
*Catalysis Group*



**Johanna Rothman**  
*Rothman Consulting  
Group, Inc.*



**Esther Derby**  
*Esther Derby Associates, Inc.*

**NEW FOR 2010  
COLLOCATED  
CONFERENCES!**



***Your registration  
includes full access to  
the Agile Development  
Practices|West conference!***

Conference  
Sponsor:



Industry Sponsors:





# Build Your Conference!

Conference schedule includes multi-day training classes, tutorials, keynote presentations, concurrent sessions, summit sessions, and more!

**BETTER  
SOFTWARE**  
**CONFERENCE**

## SUNDAY

Software Tester Certification—Foundation Level Training (3 days)  
Scrum Master Certification (3 days)  
Practical Test-driven Development (3 days)

Agile Testing Practices (2 days)  
The Foundations of Agile Development

## MONDAY – TUESDAY

Choose from 37 half-and full-day tutorials that allow you to learn in-depth about specific topics.

### Popular Tutorials Include:

Practical Software Measurement: Information for Decision Makers  
Project Assessments: Knowing Where You Stand  
Test Estimation for Development and Test Managers  
Adrenaline Junkies and Template Zombies  
Risk-based Testing: A Systematic Approach

The Leadership Tutorial: Improving Your Ability to Stand and Deliver  
Project Risk Management: A Systematic Approach  
Release Planning: A Strategy for Success  
Bridging the Gap Between Management and Test  
Fostering Trust in Teams: A Leadership Practicum

## WEDNESDAY – THURSDAY

4 Keynote Presentations  
24 Concurrent Sessions  
Networking EXPO  
Special Events  
...and More!



Leading Projects & Teams  
Business Analysis & Requirements

Processes & Metrics  
Testing & QA

Plan-driven development  
And Much More!

## FRIDAY

### Agile Leadership Summit

Add a fifth day to your conference event by attending the Agile Leadership Summit—Taking Agility to the Next Level. Join your peers and agile industry veterans to explore the unique challenges facing software development leaders as agile practices move into the mainstream. You'll hear what's working—and not working—and have the opportunity to share your experiences and successes.



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

REGISTER BY APRIL 9, 2010 AND SAVE UP TO \$400!

# Selling to Your Buyer

Consider the needs of the buyer as stakeholder. When you have no buyers, you have no users.

by **Scott Sehlhorst** | [scott@tynerblain.com](mailto:scott@tynerblain.com)

One of the biggest challenges you face as a business analyst is identifying who your product needs to satisfy. You already know to satisfy your “stakeholders,” but who are they? You’ve already thought about your users, who have real, valuable problems to solve; they are definitely stakeholders. But the people who make buying decisions are also stakeholders, and they have their own set of problems. If you don’t solve their problems, your product will never get in the hands of your users, because the buyers won’t buy it for them.

## What It Takes to Succeed

If you view product creation and sale as a process, you will see that there are three key stages to success: you have to create the product, the product has to solve a valuable problem for your customers, and you have to sell the product.

Most business analysts are acutely aware of the impact of requirements on solving the user’s problems. Most good business analysts also realize the benefits of defining requirements that make it cost effective to create the product. What many business analysts overlook, however, is making sure that the product can be sold. If you don’t have customers, then you don’t have users, and nobody wins.

## Buyers Versus Users

When you’re selling consumer products, your user often is also your buyer, but not always. For example, this clearly is not the case when the products are targeted at children.

You can develop a toy for three- to five-year-olds that may address all of the entertainment, safety, and development needs of your toddler users, exciting them to no end and creating an irrepressible demand. However, if you don’t conspicuously address the needs of their parents, the parents will not purchase your toy, and your target toddlers will never get to experience the joys of playing with it.

Sales of commercial products and business-to-business sales operate the same way. You may be developing requirements for users of a content management system, but unless you can convince the executive stakeholders who have the budget and the purchasing agents who approve purchases, your target users will never get a chance to benefit from the capabilities you’ve built into your product.

The key to assuring that you can sell your product is in understanding the goals of the buyers. The best way to develop and communicate an understanding of the buyers is through the development of buyer personas.

## Buyer Personas

Like user personas, buyer personas are archetypes that represent people with shared problems, perspectives, and objectives. There are, however, key differences between buyer personas and user personas in the perspectives and problems that buyers and users face.

To develop a buyer persona, you must start by understanding your buyer’s goal: The buyer is trying to solve the user’s problems by selecting a product that will solve what she (the buyer) believes to be the problem. The buyer persona is a description of the buyer—how she thinks about the problems that the users face, how the buyer will make tradeoffs in purchase decisions (save now versus save later), and what the buyer believes are the most important problems to be solved for the users. In your description, capture how you believe the buyer will approach the buying process—comparison of objective facts about the solutions or emotional, impulsive decisions. Also, determine if the buyer will focus on obvious costs—deployment and technology challenges—or the ultimate utility that (she believes) the users will receive from the product.

Is the buyer someone who is convinced by anecdotal arguments or statistics and trends? Provide both, but emphasize the one that is most effective for the buyer.

To develop a product that meets the needs of a user persona, you have to address the user’s problems. To develop a product that meets the needs of a buyer persona, you have to address the buyer’s understanding of the user’s problems, as well as any other problems on which the buyer—but not the user—may be focused (e.g., costs and deployment challenges).

## First Impressions

Imagine that you are mingling at a party and you are introduced to someone. You quickly form a first impression of that person and, assuming things look promising, you spend some time getting to know him or her. If things go well, you go on a first date and then decide if you want to start a relationship or not.

“The key to assuring that you can sell your product is in understanding the goals of the buyers.”



Now, imagine that your friend is trying to fix you up. Your friend believes that she knows what you need and filters candidates based on her perceptions of who would be a good match for you. Have you ever been fixed up on a blind date only to ask your friend later, “What were you thinking?” Think how many viable suitors she passed over to set you up with a walking train wreck.

When your team is developing a product, it is the same as if your company were one of the suitors. When you’re selling a product where the buyer and the user are not the same person, you’re being interviewed by the matchmaker. If you don’t address the buyer’s *understanding* about what is important, you’ll never get the first “date” and never get the chance to form that perfect relationship with the user.

One way to succeed in the “matchmaker” interview is to work with the buyer to develop a list of concerns and then address them individually. Each concern will have a unique resolution or mitigation. You’re simultaneously showing how your product “scores” against this improved model and helping the buyer improve her mental model, so that the decision process is a “better one.”

## Solving Buyer Problems

You create a product to solve a user’s problem. Your requirements-definition activities center around developing an understanding of that problem and clarifying the criteria for objectively determining if your not-yet-developed product will solve it. You need to take the same approach with your buyer. If the buyer is focused on balancing capital expenditures (versus operating expenses), then offer a pricing structure that matches the buyer’s goals. If the buyer is concerned about the change-management aspects of adapting her business practices to leverage your product’s benefits, offer deployment, consulting, and training services.

The buyer does not have a problem until your user tells her that a problem needs to be solved. Your product solves a user’s problem. Purchasing your product solves the buyer’s problem.

The buyer will have a mental model of her understanding of what it takes to solve her customer’s (your prospective user’s) problem. If your product does not match the buyer’s mental model, you won’t sell your product and your user’s problem will be solved by someone else or, worse, go unsolved.

Buyers usually are not experts on the problems that their customers face. For buyers to develop a mental model of what makes a candidate product a good match, they have to put together a checklist. That checklist sometimes takes the form of a request for proposal (RFP) or request for quote (RFQ). Within an RFP or RFQ, there typically is a series of questions that is used not only to rule out products but also to provide a framework for comparison of competing products.

You have to understand what the buyer’s checklists look like and address her concerns. Make sure to prioritize the solutions that address these buying criteria along with the solutions that address real user problems. When the buyer is concerned about the viability of your company, offer under non-disclosure to share information about your company’s financial performance. For technical issues, get the buyer to designate someone in her company who can make an assessment of the pros and cons of your solution relative to each particular issue. If the buyer is concerned about your commitment to a particular industry, offer to set up a conversation with someone from your company who can demonstrate the importance of that industry to your strategy.

To create a product that is successful, it is not enough to solve only the problems of your users; you also must address the problems that the buyers perceive that your users have. If you can’t convince the buyers that your product is a match, you’ll never get a chance to deploy it to your users who are sure to love it. {end}

Visit [StickyMinds.com](http://StickyMinds.com)  
to comment on this article

## Security Analysis

### Do you have Path Insensitive Insecurity?



## McCabe IQ

**Vulnerability Analysis May Be  
the Answer**

**Path Insensitive Insecurity**, *noun*,  
\ˈpɑːθ\ in-sen(t)-s(ə)-tɪv\ in-si -kyur-ə-tē:  
a software security disease that occurs when  
programmers, designers, and software security  
analysts are not cognizant of paths within their  
source code or design that may lead to or be part  
of a known or unknown exploitable software  
vulnerability.

**McCabe IQ helps to eliminate path insensitive  
software insecurities.** The key to our  
uniqueness and the key to finding vulnerabilities  
is paths.

When designing functions and protocols, you  
should provide as few run-time options as  
possible to keep the amount of code exposed to  
attackers to a minimum. A lot of security flaws  
turn up in unused options and rarely-executed  
code. Commonly used code paths are routinely  
tested by end users, while obscure functions  
receive little attention.

Security breaches are often a result of  
interactions in software that, on the surface,  
appear innocent. Attackers can disrupt a  
system or defeat its security goals by exercising  
sequences of interdependent decisions to  
produce unforeseen, and possibly disastrous,  
consequences and unexpected results.

**Structural Security Analysis with McCabe IQ**  
can help uncover serious security flaws in code  
by analyzing control flow paths and subtree  
structures and verifying control flow integrity.

As part of a secure, trustworthy software  
development process, McCabe IQ helps  
identify and exercise paths through the code  
to ensure that program behavior is correct and  
expected. Traditional techniques for line and  
branch coverage leave too many security gaps.  
If security is important for your systems, you  
need a comprehensive validation approach that  
includes cyclomatic complexity and basis path  
analysis to scrutinize risky code structures using  
data and control flows. **You need McCabe IQ.**

Download “*Path Insensitive Insecurity*”  
at [security.mccabe.com/bettersoftware](http://security.mccabe.com/bettersoftware),  
or call 800-638-6316 to schedule a live demo.



**McCabe**  
SOFTWARE

# Experience the Advantages of Self-Paced eLearning



**Try our New Demos Now Available online  
at <http://www.sqetraining.com/eLearning>**

## New Courses Now Available in eLearning Format

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

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

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

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

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

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

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

## What are the benefits of eLearning?

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



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

# Hidden Messages

An experienced test manager can gather useful information by looking at more than the defect management system's data.

by **Dan Minkin** | [dan.a.minkin@gmail.com](mailto:dan.a.minkin@gmail.com)

As a defect manager or, more specifically, as a test manager in charge of defects, I was never so popular as when I started to use a whiteboard to keep up to date with the latest found and fixed defects. Suddenly, senior program management knew who I was and what I did. Every day they'd ask how many new defects were raised, what the priorities were, which ones were critical, and even who was raising them—data I could readily provide from my whiteboard. As the defect numbers went up, there would be more concern; as they came down, less. But as time passed, I realized that a lot of the useful information that was found on the defect management system couldn't be reported using metrics. These were the hidden messages that were important to decipher.

## Defect Description and Developer Comments

Much of what turned out to be useful information was found in the defect description and consequent developer comments. What mattered was how the description was written, not what the information was. While some testers reported defects in step-by-step detail—including preconditions and all relevant data—others simply stated that the system had crashed or that a calculation was wrong.

The large variety in reporting styles told me a number of things. It told me who were the experienced and confident testers and who were not so confident. It told me which technical areas caused frustration—which became apparent in the language (“Surely the developers can see...”, “This is the third time...”). This information added to the weight of evidence that certain functional areas were high risk.

Also, the developers' replies and consequent debate told me something about the relationship between the developers and testers. Again, the language defined much of this, providing some clear evidence that relations weren't as good as they could have been. The number of replies—often three, four, or five from each side—indicated that either the defect resolution process wasn't working and that improved contact between development and the test team was needed, or that there was some confusion about what the requirements were.

## Defect History

A defect's history sometimes gave me interesting informa-

tion. Once, when trying to track down why a given defect had been closed months before with no explanation, it was found that on the day of closing, the defect had gone through a number of hands. Following through with the parties in question led to the discovery that a wrong assumption had closed this and other defects. The defects were reinstated. On another occasion, we tracked another group of defects that had been allocated to an incorrect party. Again, studying the history enabled us to rectify incorrect details and feed lost defects back into the workflow.

**“I realized that a lot of the useful information that was found on the defect management system couldn't be reported using metrics.”**

## Defect Fields and Formats

Prior to taking a holiday, I prepared a handover note for the defect management process. During the preparation, I realized how complex the defect management system was becoming. The number of different defect statuses should have given this away. There were seventeen. There were also twenty fields for testers to fill in. A common reason we cited for why testing took

much longer than expected was because having to raise and detail so many defects was an onerous task. However, it didn't occur to me until this point that this was partly of our own making and not just a result of defect numbers.

I resolved to try to reduce this complexity, eliminating some of the mandatory fields, reducing the number of field values, and removing some of the statuses. This had a small but important effect on the speed of the defect process. Though we accepted that capturing the correct data up front was time consuming, it did mean less time was spent in the number of informal queries returning from developers.

I discovered a useful piece of information when I asked why the test reference field in particular was rarely being used. It turned out that many of the defects were found using an exploratory approach rather than following scripts. This in turn indicated that there was a problem with the quality of the supplier testing. The supplier was using our test cases to test its software (in a rigid and incorrect fashion) and at no point was experimenting or trying to break the system. This turned out to be one of the key discoveries of the project. As a result, two further intermediate stages of testing were inserted, first into the supplier test cycle and then into the user testing (a confidence test immediately after delivery).



## The Defect Tool and Its Use

As the program moved later into the testing cycles, the use of the tool became more widespread and higher profile. Who was using the tool, when, and why gave me insight into the program dynamics and events.

The program manager would ask me what the latest figures were and explain to me why he needed to know. By knowing his reason, I was able to add extra information or pinpoint defects as the situation required.

By asking myself who was interested in the information, it became possible to identify and build up relations with key players in the program. It became clear that one of the strand coordinators, who reported to the program manager, was really one of the driving forces of the program. The relationship that was fostered was mutually beneficial, with the coordinator having readily available information on software quality and the test team having an active and vociferous ally.

## Defect Statistics

The statistics that I provided on a daily basis were simple. My daily report explained how many new defects had been raised, the total number of defects to date, and how many defects existed with our supplier. No one ever asked me for historical data or how many defects had been fixed by our supplier since the start of the program or the average time it took to fix a high-priority bug.

It was just as well. I couldn't have answered any of those questions, anyway. I hadn't set up the defect system to capture that kind of information. Once a defect was fixed, I wasn't interested in it anymore. But, that raised questions in my mind. Should I have planned for the time when the questions would be asked? Was this program simply not interested in carrying forward statistics into future programs? When I questioned the program test manager, his view was firmly the latter. As a matter of pragmatism, this issue dropped into the background.

The defect management tool and process should be a guide. Statistics and defects are useful, but they are only a window onto the health of the program. As always, it is the awareness into the human element of any part of the program that makes for the full story. Elements to be aware of include the developer's writing style, the amount of "debate" within the defect descriptions, and how successfully the tool is being used. A successful test manager should not take data and information merely at face value but should use this to inform his view and to ask himself further questions. So, when you are looking at statistics from your defect management tool, know that there is more you can understand. **{end}**

*This article originally appeared on StickyMinds.com.*

Visit [StickyMinds.com](http://StickyMinds.com)  
to comment on this article

# Are you struggling to manage maintenance and development in parallel?

NEW

Get integrated release planning, task tracking and source control to help solve your development challenges.

Mapping your release planning to your project configuration allows you to increase flexibility and reduce the risk of human error. Get more than just a link between issues and code changes: get true integration.

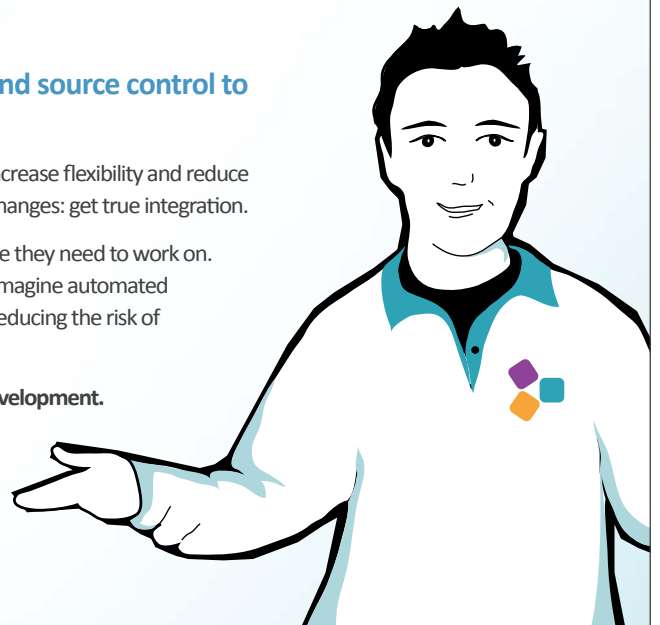
Imagine automatically populated developer workspaces based on the release they need to work on. Imagine real-time status updates throughout the development lifecycle. Or imagine automated notifications about fixes missing in a current or future release, dramatically reducing the risk of regression bugs.

Learn how PureCM Professional can help you to manage your software development.

Get the free evaluation resources at [purecm.com/start10-1](http://purecm.com/start10-1)



Get your **FREE**  
trial now at  
[purecm.com/start10-1](http://purecm.com/start10-1)

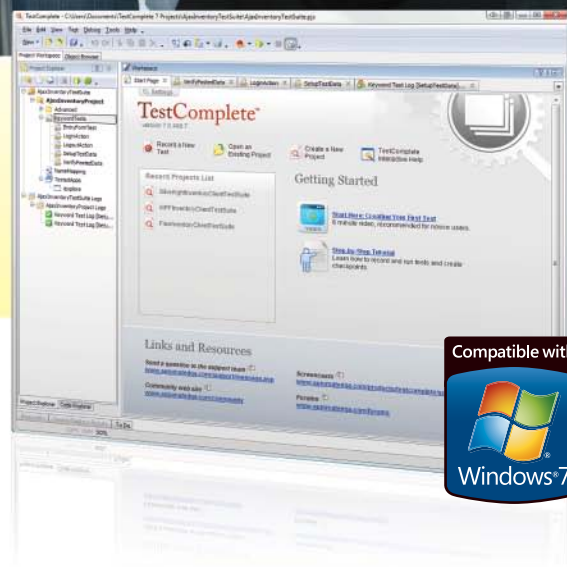




*Can you find 7 mistakes in this photo?*

*Did you find them all? Answers at [www.testcomplete.com/hawkeye](http://www.testcomplete.com/hawkeye)*

Fun challenge, right? Great software testing feels like that.  
Now do that 5 times a day, every day, for the next year...  
With the Exact. Same. Photo. Does your testing feel like that?  
Time for TestComplete.



# TestComplete™

Test faster. Test better. Test more.

Free yourself to do the quality work that you love: explore, investigate and discover.

## TEST WITH THE BEST

TestComplete is the most exciting software test automation tool available.

## TEST ANYTHING

- Web, Ajax, Flash, Flex
- Windows, .NET, WPF, Silverlight
- Java, Qt, Delphi, MS Office
- Functional, Load, Regression, Manual

## TEST IT YOUR WAY

- Automate without programming via Keyword Testing
- Free extension downloads
- Professional IDE for test engineers
- Friendly and responsive support

AutomatedQA  
Call Us: +1 (978) 236-7900



*Did you find  
all 7 mistakes?*

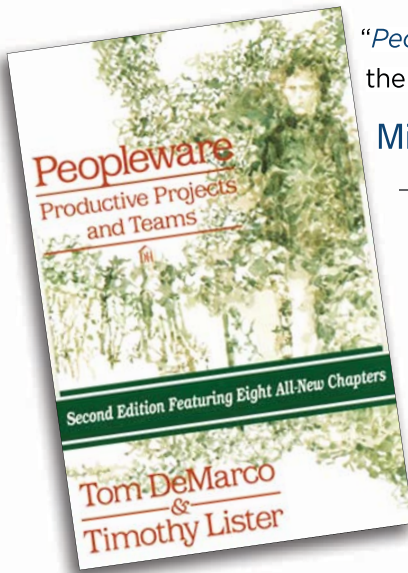
Visit [www.testcomplete.com/hawkeye](http://www.testcomplete.com/hawkeye)  
to see how you did and try TestComplete



# Virtual **Resource** Shelf

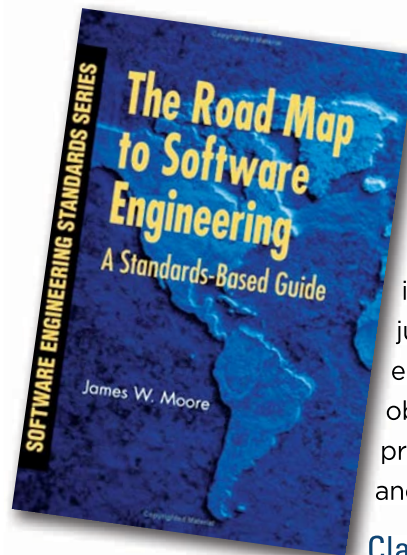
Author recommended books, blogs, gadgets, Web sites, and other tools for building better software

**Q:** What book do you consider a “must read” for someone just starting out in the software industry?



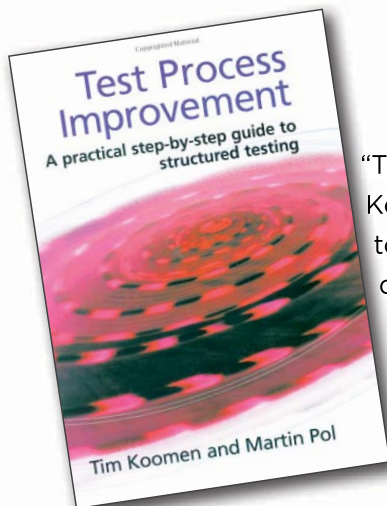
“*Peopleware* by Tom DeMarco and Timothy Lister because it’s a useful book on the importance of teamwork in delivering software products.”

Mike Cohn



“James W. Moore’s *The Road Map to Software Engineering: A Standards-Based Guide* covers all of the basic issues and points the reader to where there is detailed guidance on just about every software engineering topic. Testers can obtain consensus-reviewed best practices for both testing itself and for any form of test basis.”

Claire Lohr



“The first testing book I owned and the one I still carry around is Tim Koomen and Martin Pol’s *Test Process Improvement*. It is portable, easy to read, wide ranging, and full of suggestions. Other books are more in depth but few are as practical.”

Dan Minkin

For someone starting out as a business analyst in the software space, the only place to start is Karl Wiegers’ *Software Requirements*. Karl’s coverage is pretty comprehensive and provides both a big-picture perspective and immediately useful, detailed help for anyone writing requirements.

Scott Sehlhorst





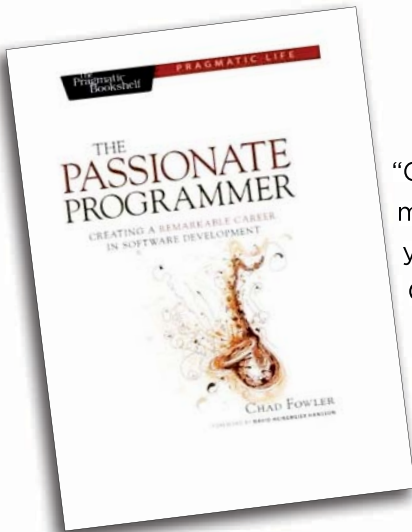
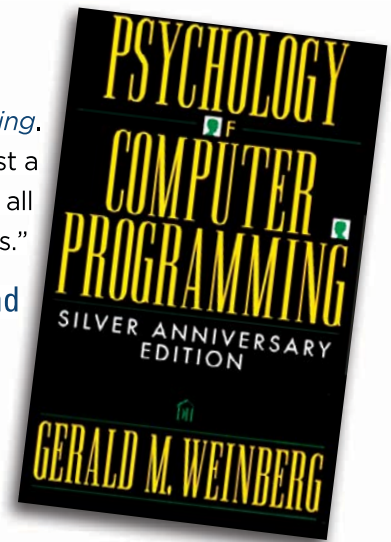
# Virtual **Resource** Shelf

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

“Gerald Weinberg’s *Psychology of Computer Programming*.

It was the first book to describe programming as not just a matter of hardware and software, but a human activity with all the richness that such activities possess.”

Lee Copeland

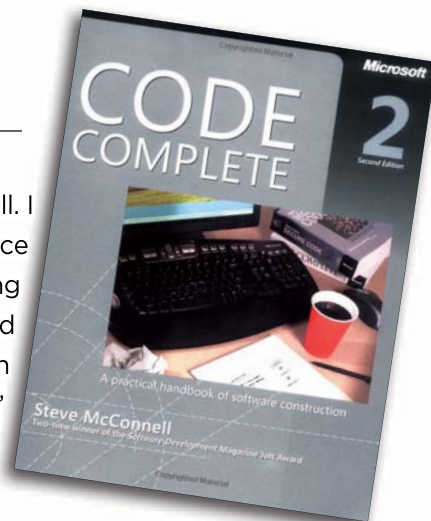


“Chad Fowler’s *The Passionate Programmer*. It doesn’t matter if you are a developer, tester, writer, or whatever. If you are starting out (or if you’ve been working for a while), Chad has great ideas about how to take advantage of all opportunities in your career.”

Johanna Rothman

“*Code Complete* by Steve McConnell. I recommend this because it gives a nice overview of the nuts and bolts of creating software. It’s essentially a handbook, and helps set the bar for someone starting out in their career as a coder.”

Jonathan Kohl



**Want to get our authors' take on a particular resource? Email us at [editors@bettersoftware.com](mailto:editors@bettersoftware.com) Subject: **Virtual Resource Shelf**.**

# FUZZING and the SDL

**The increased complexity of new technologies and faster software release cycles are making signature-based security solutions redundant. Instead more adaptable preemptive testing is needed. Moreover, due to outsourcing, the software development process itself is also becoming more complicated creating the need for more powerful project management tools such as software development lifecycle models (SDLC). The Security Development Lifecycle (SDL) combines preemptive security testing with the SDLC model.**

## The Microsoft SDL

The Microsoft SDL is the industry-leading software security development process. It is founded on the idea that security should be built into software during the software development process. It basically lists all the security measures that need to be carried out during the development process grouped according to different phases of the SDLC. For testers the most important phase in the SDL is the verification phase, where fuzzing is used to test the robustness of the system before the release.

## Fuzzing in the SDL

Fuzzing is a natural part of the SDL, because the entire fuzzing methodology promotes building security into systems, instead of protecting vulnerable systems. In the Microsoft SDL, fuzzing is used in the Verification phase. However, fuzzing can be used

throughout the development process from the moment the first software components are ready and even after the release. The earlier the vulnerabilities are found, the easier and cheaper it is to fix them.

The idea behind fuzzing is simple: unexpected data is fed to the inputs of a system and the behavior of the system is then monitored. If the system crashes, then there is a bug in the software, which could have been exploited by attackers. However, the vulnerabilities can also be triggered by simple unexpected inputs in events like heavier than normal use, system maintenance or when other devices or systems send malformed inputs as a result of error conditions. A zero-day vulnerability is a ticking time bomb, it does not really matter what eventually sets it off. The important thing is to get rid of them.

Codenomicon was selected to participate as a tool vendor in the Microsoft SDL Pro Network, which assists companies in integrating the SDL into their development processes. Its off-the-shelf Defensics solutions provide an easy way to fulfill the SDL's fuzzing requirement. No test tool creation or maintenance effort is needed, and as a software based solution Defensics is easy to integrate into existing tool libraries.

## Answering testing challenges

The biggest challenges in Fuzzing are handling the infinite number of possible test cases and reaching the deeper protocol layers. Codenomicon's intelligent fuzzing methods, which are based on protocol specifications, cover the entire protocol implementation, and thus they can target protocol

areas most susceptible to vulnerabilities to shorten test run times. Robustness tests can be run, for example, automatically with every build. Intelligent Fuzzing tests can also function as the peer of the tested system and go through extensive message sequences making it possible to identify vulnerabilities in deeper protocol layers.

With new technologies the specifications are not always available. In such cases, traffic captures can be used to create mutation-based fuzzers. Traffic capture fuzzers are based on real traffic samples, thus they only cover a limited selection of the implementation. Yet, performing traffic capture based tests already significantly improves the security and robustness of the system. Moreover, no additional knowledge is needed, thus traffic capture based tests are quick to create and easy to execute. Codenomicon's traffic capture fuzzer is an excellent tool for getting started with zero-day bug discovery.

## Benefits of Testing with Defensics™

The biggest benefit of Codenomicon's fuzzing method is its ability to find not only known, but also unknown or zero-day vulnerabilities. Fuzzing is essentially simulating attacks against a system under test to find vulnerabilities in it. Firewalls and virus scanners only look for known vulnerabilities and they add to the complexity of your system increasing its attack surface. Furthermore, application service providers, like online-banks and web stores, cannot build fortresses around their application, because their customers need to be able to use their services.

Gain time and save money through preemptive testing. Instead of racing against the clock to get your system to work again and spending valuable resources in calming down angry customers, use fuzzing to discover flaws and create patches for them proactively, before any problems occur. By integrating fuzz testing into your software development lifecycle, you can discover flaws at the earliest possible moment and save valuable resources.

*To read more about Codenomicon and integrating fuzzing into your software development lifecycle, go to <http://www.codenomicon.com/bettersoftware/>*

## FREE FUZZING TOOL!

Unique opportunity to test top-of-the-range commercial testing software for free.

**Request your free copy now!**

<http://www.codenomicon.com/free-ftp-suite/>



- 1 What do you consider the biggest benefit to cloud computing?
- 2 What is your main concern with cloud computing?
- 3 What percentage of your organization's IT resources have moved to the cloud?

To participate in the survey, go to [www.stickyminds.com/cloudsurvey](http://www.stickyminds.com/cloudsurvey). Results will appear in *Better Software* magazine's May/June 2010 print and digital editions.



**You'll spend \$15 on a pizza, so why not on an annual subscription to *Better Software* magazine?**

**For an entire year!**

**Only \$14.95!**

**BETTER SOFTWARE MAGAZINE**

Take advantage of our \$14.95 digital to print upgrade and be entered in a drawing for one of fifteen chances to have a pizza delivered to your office compliments of *Better Software* magazine by April 30, 2010.

To upgrade your subscription and enter for a chance to win  
[www.bettersoftware.com/pizza](http://www.bettersoftware.com/pizza)

To enter without subscribing  
<http://forms.sqe.com/forms/MKP-BSMWinaPizza>





Conference Sponsor:  
 **RALLY**  
SOFTWARE

**June 6-11, 2010**  
**Las Vegas, NV**

 **SOFTWARE  
QUALITY  
ENGINEERING**  
**CONFERENCES**

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

REGISTER BY APRIL 9, 2010 AND  
**SAVE UP TO \$400**  
GROUPS OF 2 SAVE EVEN MORE!

## KEYNOTES

BY INTERNATIONAL EXPERTS



**Tim Lister**  
*Atlantic Systems Guild*



**Robert C. Martin**  
*Object Mentor*



**Johanna Rothman**  
*Rothman Consulting  
Group, Inc.*



**Ryan Martens**  
*Rally Software Development*

**NEW FOR 2010**  
**COLLOCATED**  
**CONFERENCES!**



+



***Your registration  
includes full access to  
the Better Software  
Conference!***

Conference  
Sponsor:



Industry Sponsors:



*Net* **Objectives**



# Build Your Conference!

Conference schedule includes multi-day training classes, tutorials, keynote presentations, concurrent sessions, summit sessions, and more!



## SUNDAY

Software Tester Certification—Foundation Level Training (3 days)  
Scrum Master Certification (3 days)  
Practical Test-driven Development (3 days)

Agile Testing Practices (2 days)  
The Foundations of Agile Development

## MONDAY – TUESDAY

Choose from 37 half-and full-day tutorials that allow you to learn in-depth about specific topics.

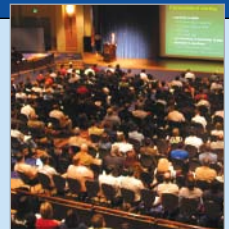
### Popular Tutorials Include:

ADAPtIng to Agile: A Guide to Transitioning  
Ensure Project Success with Scrum—The Right Way  
Releasing Large-Scale Agile Projects  
Getting Agile with Scrum  
Writing Effective Agile Use Cases

Agile Release Planning, Metrics, and Retrospectives  
Systems Thinking for Lean and Agile Growth  
Agile Estimating and Planning  
Principles and Practices of Lean-Agile Development  
Agile Project Planning: Building Strong Backlogs

## WEDNESDAY – THURSDAY

4 Keynote Presentations  
24 Concurrent Sessions  
Networking EXPO  
Special Events  
...and More!



Transitioning to Agile  
Agile Implementation

Agile Development Techniques  
Agile Testing

Scrum  
And Much More!

## FRIDAY

### Agile Leadership Summit

Add a fifth day to your conference event by attending the Agile Leadership Summit—Taking Agility to the Next Level. Join your peers and agile industry veterans to explore the unique challenges facing software development leaders as agile practices move into the mainstream. You'll hear what's working—and not working—and have the opportunity to share your experiences and successes.





# DEMYSTIFYING EXPLORATORY TESTING

BY JONATHAN KOHL



**E**ARLY IN MY TESTING CAREER, I discovered exploratory testing (simultaneous test design, execution, and learning [1]) and tried to practice what experts recommended in their articles. I thought I was applying their ideas as best I could, and I knew what I was doing would be called “exploratory testing,” but I was concerned. I wondered if I was missing something. I had this gnawing fear that I had misinterpreted or misunderstood a key concept. In short, I lacked confidence in my exploratory testing approach.

You can imagine my surprise when I met exploratory testing trainers who were more interested in *my* ideas and experiences than in my ability to recite theirs. Now that I mentor and train exploratory testers, I find that I, too, am more interested in other testers’ unique practices, ideas, and experiences than my own. That’s what’s wonderful about thinking about testing from an exploratory perspective: It is human centered. Instead of dehumanizing the tester by reducing testing to a clerical task or by trying to automate away testers, we embrace their unique skills and experiences and the value they add to our projects. We use both software and thinking tools to give testers more power, not diminish it.

Recently, a test manager approached me after an exploratory testing training course. “Finally, I have words to describe what my team and I have been doing for years!” she said and thanked me for not making them feel stupid for being unconventional in their testing work. “You’re the first consultant I’ve talked to who didn’t tell me I was doing it wrong.” She wasn’t “doing testing wrong”—in fact, the development team was delighted with the results of her team’s testing efforts. Trouble was, the consultants who had worked with her team were uncomfortable with her exploratory testing approach and the testing styles her team typically used. This is an important lesson: Just because an approach is unconventional doesn’t mean it’s wrong.

## Exploratory Testing Styles

Because exploratory testing has less visible structure—for example, it lacks explicit steps and expected results—the test execution is different depending on who is doing the testing. There are different styles and variations that often yield similar results. What follows are some styles I’ve observed.

### INTUITIVE

This is the most common style. Testers who haven’t learned specific exploratory testing techniques tend to do this naturally. When you ask them what they are doing when they are testing in the absence of pre-scripted test cases, they may say, “I don’t know why I did that,” or that they are using their intuition. Intuition is just a fancy way of saying, “I am doing this because of the insight I have based on my experience and knowledge.” It can appear to be random or chaotic, but when the tester is pressed for an explanation of what he did, a structure and purpose emerge.

### LEARNING DOMINANT

This is also very common. Even shops that insist that all their test cases be designed and recorded prior to formal testing use this exploratory testing style during test design. While writing test cases, the test designers will try out new software or a new feature that they need to learn about. As they learn, they invariably find bugs. This is also true of automated testing. As the automators learn about the system, they often will use an exploratory approach as they try to get the automation tool to interact correctly with the application.

The concept of *touring* [2] heuristics to learn about applications is becoming popular. Touring is a good test idea-generation activity. Testers create models of the software they are testing or use mnemonics to help them look at the application in different ways. This learning about the application leads to bug discoveries and test ideas. Some examples include creating coverage maps and following them or using different kinds of touring in a heuristic. For example, James Bach talks about touring the application to see if you can identify all the features. Or, try using the software the way you imagine different users would. Touring is a powerful tool for test idea-generation and discovery.

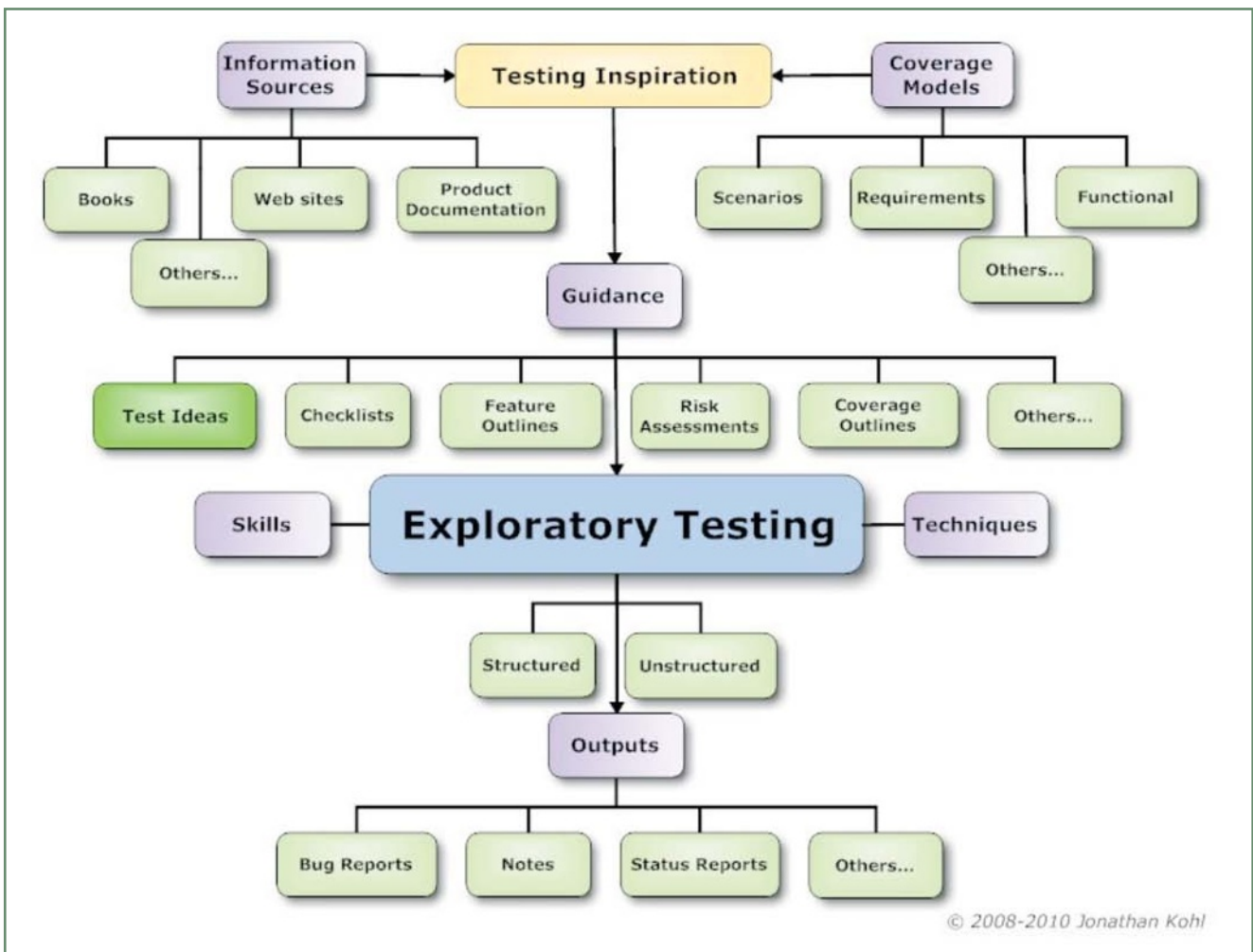


Figure 1: Exploratory testing mind map

## SYSTEMATIC

More advanced exploratory testing involves using a particular system, model, or strategy to guide your testing. This is done to be more thorough in looking at and testing the system, and to be more consistent in our exploratory testing work. When I started working with James Bach, one thing he noticed was inconsistency in my exploratory testing execution. I might use one set of test ideas and tools on one application but a different set on another. He taught me to use abstract models (see his Heuristic Test Strategy Model for more information) during test execution to guide my thinking and to help my exploratory testing be more repeatable and consistent.

## REGRESSION

Yes, even exploratory testers have to do regression testing. The difference between what scripted testers think of as regression testing and the exploratory testers' version is in the level of guidance used. Exploratory testers often use lightweight coverage outlines as guidance rather than test cases. The guidance of the outlines helps ensure we can easily and frequently repeat testing in desired areas. Using lightweight guidance with an exploratory approach to regression testing can be quite effective. Stakeholders have a good idea of what

is being covered, such as features, functions, user scenarios, etc., while still allowing the test executors' natural variation to discover problems quickly.

## INTERACTIVE AUTOMATED/TOOL SUPPORTED [3, 4]

Exploratory testers often use automation tools to help them. For example, if regression testing is becoming a burden, some aspects of testing may be automated. Task automation focuses on speeding up activities like data loads or installing new builds and can extend to test setup for exploratory testing sessions. [5] These tests are run under the supervision and direct control of the tester who can step in and intervene manually at any time.

Exploratory testers also use test automation that runs unattended, particularly if the tasks are better served by a machine than a human. The interactive part comes in when exploratory testers use the tools to help them simulate different kinds of conditions and then observe the results, change factors, and rerun the tests.

## Enough Words! How about a Picture?

Most people who understand the unscripted, improvisational sense of exploratory testing but worry whether they are

doing it right aren't necessarily doing anything wrong; they merely lack confidence. One way to banish a lack of confidence is to develop your testing skills. Describing exploratory testing can be difficult, but I've found the mind map in figure 1 to be helpful. As you look through the mind map, does it help your thinking about exploratory testing? Can you identify areas that resonate with you?

I use this mind map to show that, although there may be little visible evidence, exploratory testing still can be a disciplined, structured form of testing. It's just that the tester keeps track of more of the structure and guidance in his head. Exploratory testing isn't just "playing with a product until it breaks," and it doesn't need to be a set of simple "quick tests." Often improperly characterized as "superficial bug hunting" or "quicktests" [6], exploratory testing can, in fact, take us far beyond that.

## Coverage Models

When testing, we have some idea of coverage, or how much we are going to test a product. Cem Kaner defines coverage as "... the amount of testing done of a certain type." [7] We can expand that to *the amount we are testing software using different approaches*. We reveal different information if we test an application in different areas, in different ways, or with different techniques and tools.

There are many ways of defining coverage [8], which are called *coverage models*. A coverage model describes how you are going to test the software. Because testing from different perspectives reveals different kinds of information, it's better to use different kinds of coverage models than one single model. [9] For example, we could use a requirements-based approach and define a certain amount of testing to verify the software meets requirements. We could also determine coverage using different testing techniques, such as defining coverage according to security, performance, accessibility, etc. and defining an appropriate amount of testing in each area. We will usually use one model at a time when testing, but it is not uncommon to change models on the fly.

Choosing appropriate models can be determined by how we want to mitigate risk on a project or through other motivations. It depends on what our stakeholders are looking for. For example, with software startups, stakeholders often are more concerned with how testing information affects their ability to sell their product (capitalize on potential rewards) than in direct risk mitigation.

James Bach's SFDPOT "San Francisco Depot" mnemonic [10] (an abstract model for testing) spells out six possible models of coverage for exploratory testers: the *structure* of the application, its *function*, the *data* it uses, the *platforms* it runs on, the *operations* its users typically engage in, and the impact of *time* on the system.

One lightweight method of keeping track of what was tested and when is to use *coverage outlines*. Coverage outlines are physical documents that often take the form of checklists in a spreadsheet or visual diagrams that show user flows through the system. Their purpose is to direct and guide testing and to communicate what was tested and why. They

are defined at a higher level than test cases, spelling out products, areas to be tested, and lists of testing ideas. A tester may run one test in a checklist based on that idea, or he may run more, depending on the task at hand, his schedule, and what he discovers while testing.

## Information Sources

We use all sorts of information when testing. With experience, we grow our own testing toolbox of ideas and tools, and we sometimes find it difficult to explain where our compiled knowledge comes from. This knowledge comes from our testing experience, as well as books, Web sites, product information, subject matter experts, other team members, and conference, course, or workshop materials. It is preferable to use many rich sources of information to help guide our testing. As information providers regarding the quality of the product under test, any information that helps us do our work is useful. The richer our information sources, the more informed our testing will be.

## Guidance

When we test, we usually have some sort of guide. This can be as simple as a bug report that we use for a bug fix verification, a test idea, or a test goal or charter for a testing session. Guidance can be more formal in the form of test checklists, coverage outlines, visual coverage, feature maps, or test cases. What we use for guidance is informed both by our coverage models and our information sources. For example, we may have a coverage outline for functional testing that is informed by our own knowledge and experience, as well as by sources of information from our project and from external sources. This coverage outline will guide our testing.

Guidance also comes in the form of experienced team members and outsiders who can help you and your team. In my work with testing teams, I help them develop skills and strategies so they can accelerate their learning and reduce their trial and error. Using outside expertise such as a trainer, article, concept, or coach can help. Also, look for expertise within your team. More experienced teammates are a wealth of information and are great to bounce testing ideas off of.

## Structured vs. Unstructured

Structured exploratory testing has more guidance and more visible tools and documentation. For example, session-based test management [11] is a method for adding rigor to recording exploratory testing so it may be reviewed or audited. Instead of writing test cases in advance, testers take detailed notes while testing. These notes are collected, reviewed, and stored for audit purposes. At the other extreme, we may engage in unstructured exploratory testing, which is sometimes characterized as "ad hoc" or "jumping around the application to find problems." Exploratory testing can be very unstructured or very disciplined—and it usually falls somewhere between the two.

## Techniques

Since exploratory testing is an approach to testing, not a



technique [6], any testing technique can be utilized if it helps. Classic analysis techniques such as boundary value analysis, equivalence class partitioning, root-cause analysis, classification trees, and others can be applied to exploratory testing. If a technique helps you generate testing ideas that lead to better testing, use it. An exploratory testing approach can be used with any kind of testing technique, such as performance, load, functional, usability, accessibility, security ... the list goes on and on.

## Skills

It is difficult to narrow down a short list of skills useful for software testing. I've worked with a lot of different testers with different skill sets, and the testers tend to draw on different areas of experience. I see the following skills most often:

**Strategic thinking:** Pick an area of focus and adjust testing to provide information in areas that yield the most useful information.

**Idea generation:** Exploratory testers need to be able to think of different ways to test the software in the moment. If they see something strange, they adapt their testing. This takes practice, and to do it well, testers use mnemonics to help remember different strategies and techniques.

**Investigation:** Analyze the product and the space in which it operates, find areas of interest, and explore those areas using tools, processes, and testing techniques to reveal useful information.

**Communication:** It's important to remember what you did (if you didn't record a testing session) and explain your strategy and what problems you found. If we discover important information but can't communicate it in a variety of ways, that information doesn't help the team.

I tend to talk a lot about exploratory testing as investigation (something I picked up from Cem Kaner) and related investigational skills. Technical skills that we develop on software teams—from programming to technical writing—are a natural fit. Skills related to understanding the business, our customers, and how they use our software are also very useful. I've seen wonderful testing from former forensic accountants and stock traders. Both professions attract natural investigators who are taught to assess systems rapidly and listen to their "gut feel" or intuition.

## Outputs

When we test, we discover information that stakeholders on the team need to help make decisions about the software. We provide information to other stakeholders in many ways:

**Bug reports:** Typically distributed to the technical team, bug reports need to be well written so programmers can easily reproduce and fix the bugs.

**Notes:** Personal notes help us as we make our testing observations visible and help us create reports that can be shared with our teammates.

**Session sheets:** Descriptions of testing activities captured during session-based exploratory testing can be used in regulated environments and can be used to help review our work

with colleagues to see if we have overlooked anything. With teams that use both exploratory and scripted testing approaches, session sheets and personal notes can help in test case creation.

**Status reports:** From low-tech testing dashboards [12] to verbal reports to stakeholders, communicating status helps the rest of the team understand what we have tested and the state of the application under test.

**Test coverage reports:** Stakeholders want to know how much testing of a certain type we have completed. Historically, we have counted test cases, but using multiple models of coverage and explaining what we actually worked on during testing can provide richer information than bug counts or pass/fail metrics.

**Quality criteria:** It's important to have measures to determine whether we are making our quality commitments with our software. For example, HP has used the FURPS+ model of quality criteria. [13] When using these sorts of models, we can report the functional applicability of the system under test and the usability, reliability, performance, and support measures that we have agreed on as a team.

**Blocking issues, problem areas:** It's important to raise awareness to decision makers when we are unable to complete aspects of our work. It's also very important to provide them with a sense of our qualitative impressions of the products we are testing.

**Effort and priorities:** We can't test everything, and we are usually on a time budget, so we need to express what we have tested and what we think we need to test. We also need to demonstrate a sense of priority.

Our testing outputs are our only visible work products that other team members can see. We need to put effort into determining what others need, be accountable for our work, and provide stakeholders such as programmers, managers, and decision makers with the information they require. If the stakeholders are concerned with our approach, we need to adjust and provide them with what they need.

Ultimately, exploratory testing is determined by the mind of the tester who is doing the testing, and, as a result, it is interpreted in many different ways. Don't let fear, uncertainty, and doubt get in the way of your testing. Instead, use exploratory testing references to help guide your skill development and share your experiences with others. Sharing our ideas about testing, inviting scrutiny, and making our testing approach defensible are important. If your team appreciates the testing information you provide, don't worry that you're doing it wrong. If you are practicing your exploratory testing approach and your testing is adding value to your team's efforts, *you are doing it right.* {end}

jonathan@kohl.ca



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

■ References

# Becoming Agile in Challenging Times

## Shifts in Organizational Thinking

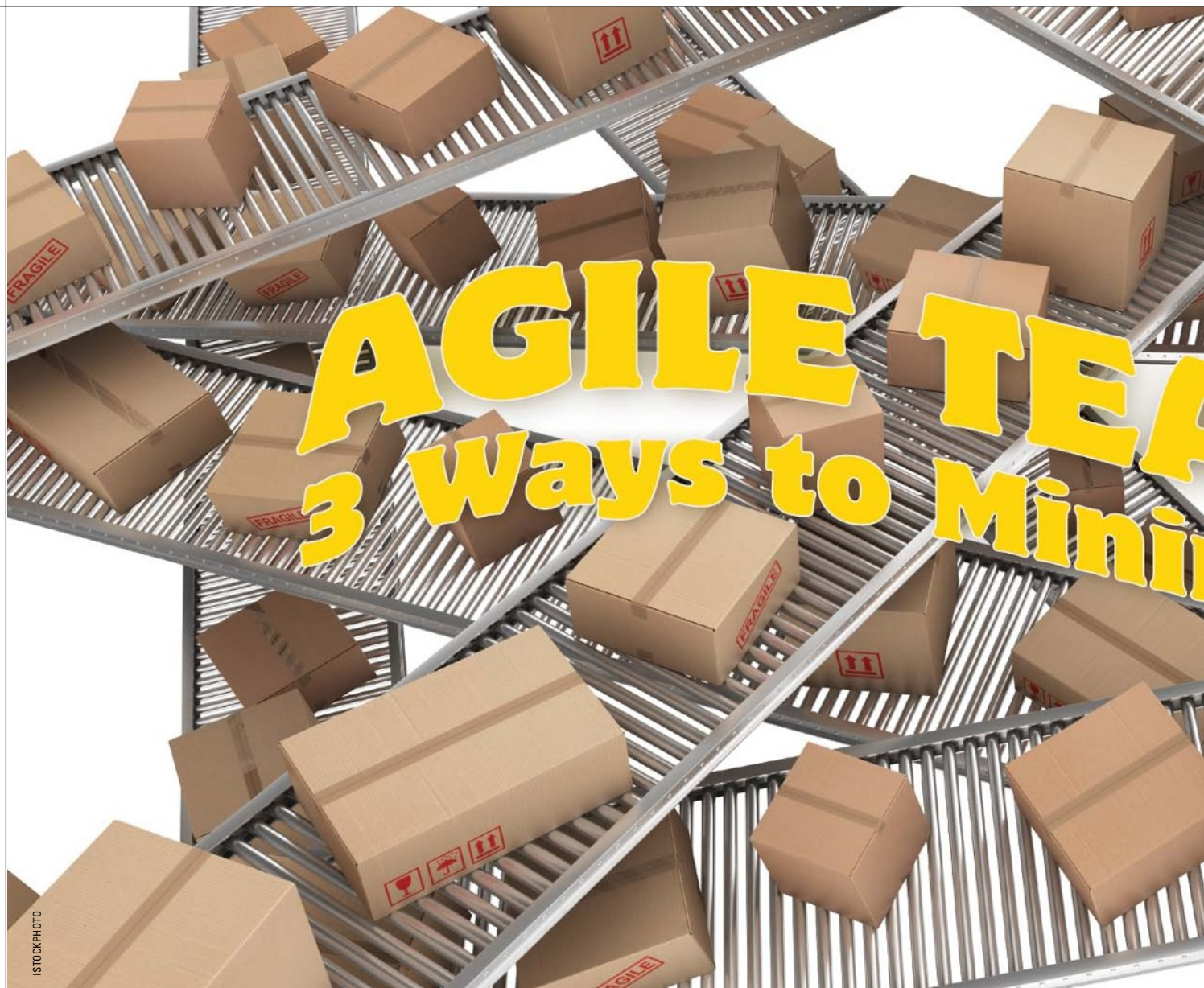
Agile provides a great acceleration method to market,  
but orchestrating the change can seem challenging.

Putting our  
imprint on  
technology  
since 1921

New approaches in project methodologies, aligning  
multiple resources, and introducing new  
technologies—all under time constraints—can be  
daunting. Join HP and Avnet to see how bringing  
together the different IT teams in an organization can  
help address concerns around less time to planning  
and less time to execution, and provide the right  
visibility into the overall status of your projects.

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





**T**eams using a sequential development process have become accustomed to handoffs between specialists. Analysts hand their work to designers, who hand it to programmers, who pass it on to testers. Teams that are new to Scrum often do not go far enough in eliminating these handoffs, wrongly assuming, for instance, that the programmers should finish programming a product backlog item before handing it off to the testers or that analysts should work at least one sprint ahead of the rest of the team.

High-performing Scrum teams, however, have learned to do a little bit of everything all the time during a sprint, thereby eliminating large handoffs. To do this effectively, teams must make three changes: favor talking over writing, make handoffs very small and very often, and mix the size of items that are brought into each sprint.

### Stop Writing and Start Talking

There's a grand myth about requirements: If you write them down, users will get exactly what they want. That's not true. At best, users will get exactly what was written down,

which may or may not be anything like what they really want. As such, Scrum teams forego a lengthy, up-front requirements phase and the resulting product specification in favor of a dynamic product backlog.

Because Scrum teams shift focus from writing about requirements to talking about them, conversations with the product owner—rather than a product spec—become the primary way of finding out how a new feature should behave. Team members talk with the product owner about how a feature should work, how quickly it should perform, what acceptance criteria must be passed, and so on. Scrum team members also talk with users, customers, and other stakeholders as appropriate and, perhaps most importantly, with each other.

On traditional projects, analysts often act as intermediaries, communicating customer needs to programmers. On Scrum teams, however, analysts function as facilitators, ensuring that appropriate discussions between team and product owner take place. For instance, an analyst might steer the product owner and team toward talking about a particular user story because it has more risk than another of going astray. At other





# AMWORK

## imize Handoffs

by Mike Cohn

times, an analyst might convey a top-level understanding of a new feature to the team before bringing the team and product owner together to discuss the details. In all cases, analysts on Scrum teams foster communication rather than distill information through a document.

All of this is not to say we should abandon written requirements documents. Rather, we should use documents where appropriate. Experienced Scrum teams lean heavily on cleanly written code and automated test cases. They then augment these forms of documentation with a written requirements document only to the extent that such a document is helpful or required for regulatory, contractual, or legal purposes. As Tom Poppendieck, coauthor of books on lean software development, once told me, “When documents are mostly to enable handoffs, they are evil. When they capture a record of a conversation that is best not forgotten, they are valuable.”

### Transfer Small Tasks Frequently

On Scrum teams, the unit of transfer between disciplines should be smaller than an individual product backlog item. That is, although there will always be some handoffs, the

amount of work being transferred from one person to the next should generally be as small as possible.

As an example, suppose a team is developing a new eCommerce application. The team chooses to work on this user story from its product backlog: “As a shopper, I can select how I want items shipped based on the actual costs of shipping to my address so that I can make the best decision.” At the beginning of the sprint, those who are interested in or who will be involved in developing this feature begin to discuss it. Let’s suppose this group includes the product owner, a business analyst, a tester, and a programmer. They begin by talking about some of the general requirements implicit in this feature: Which shipping companies (FedEx, DHL, etc.) do we support? Do we want to support overnight delivery? Two-day delivery? Three-day delivery?

As these discussions occur, the individuals involved naturally will begin thinking about how to get started. On a traditional project, each person would be able to start however he wanted (after the work was handed over). On a Scrum team, however, how to get started should be a collaborative discussion among those who will work on this feature. For this example, let’s

**“A symptom of continuing to hand off work in overly large chunks is a tendency for no product backlog items to be finished until the last few days of the sprint.”**

assume that the programmer for some reason makes the case that it will be easier to start with FedEx. The tester agrees. The analyst states an intention to investigate DHL and learn more about the parameters that affect DHL shipping costs. The analyst's goal is to have that information available by the time the programmer and tester finish with FedEx.

When the programmer knows enough to get started coding, she does so. The product owner, analyst, and tester discuss high-level tests (Will our site ship any odd-sized items like skis?). After that discussion, the tester turns the high-level list of tests into concrete tests (boxes of this size and weight going to that destination). The tester creates test data and automates the tests. Some automation may be possible without any interim deliverables from the programmer, while full automation may require getting an early version from the programmer. While the tester is thinking of the concrete tests, he also should inform the programmer of any test cases that she may not be considering while she's programming.

When the programmer and tester have finished, they add support for calculating FedEx shipping costs into the build, complete with automated tests. Graphically, this can be depicted as shown in figure 1, where four individuals work together on one product backlog item rather than handing it off to each other.

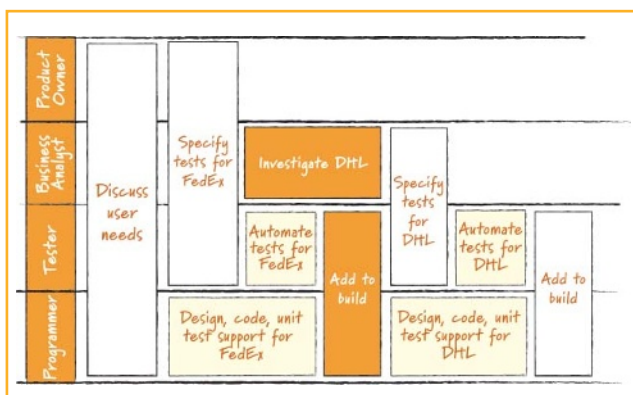


Figure 1

Next, the programmer and tester check in with the business analyst, who hopefully has learned more about calculating DHL shipping costs. The process is repeated, and support for DHL shipping calculations is added to the application when the programming and testing are complete. The key element in figure 1 is that the team has learned to work by doing a little of everything all the time. Rather than an analysis phase (done without the programmer and tester) followed by a programming phase followed by a testing phase, a little of each of those activities is happening at all times.

A symptom of continuing to hand off work in overly large chunks is a tendency for no product backlog items to be finished until the last few days of the sprint. Testers on teams that work this way often complain that they are given nothing to test until two days before the end of a sprint and then are

expected to test everything that quickly. The best way to expose this problem is to create a chart of the number of product backlog items finished as of each day in the sprint. An example can be seen in figure 2a. As the ScrumMaster on a team, I often just hang this chart in the team area with no fanfare or explanation. Team members soon figure out the problem exposed by a chart like this and, hopefully, start to find ways to finish product backlog items sooner. The result often will be similar to figure 2b, which shows a much smoother flow through the sprint.

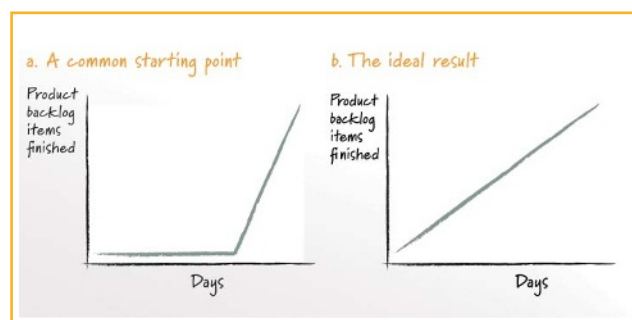


Figure 2

## Create a Manageable Mix

When planning a sprint, teams should pay attention to the sizes of the product backlog items to which they are committing. Some product backlog items are more complex than the FedEx/DHL example given. It's possible that some product backlog items might require a week or more of programming before a programmer can provide a tester with something even initially testable. That's OK. Not everything can be split as small as we might like. The key is to avoid bringing a bunch of items like this into the same sprint. Doing so will shift too much testing work to the end of the sprint.

Instead of planning a sprint with, for example, three very large items that cannot be partially implemented, bring one or two such items into the sprint along with two or three smaller items. Some of the programmers can work on the large items, handing them over to testers whenever possible. The remaining programmers can work on the smaller items, ensuring a somewhat smoother flow of work to testers early in the sprint.

Creating the right sense of teamwork can be challenging. ScrumMasters can help by ensuring that the team embraces the concept of whole-team responsibility and whole-team commitment to deliver working software at the end of each sprint. Though the team might struggle at first to break long-held habits of specialization and handoffs, increasing communication, decreasing the size of handoffs, and mixing the size of backlog items brought into a sprint will help individuals make the shift from sequential development to working as a team. {end}

mike@mountaingoatsoftware.com





**STAR  
WEST**

# SOFTWARE TESTING

ANALYSIS & REVIEW



September 26–  
October 1, 2010

San Diego, California  
Hilton San Diego Bayfront

THE GREATEST SOFTWARE TESTING CONFERENCE ON EARTH

Choose from a full week of  
learning, networking, and more

## SUNDAY

Multi-day Training Classes begin

## MONDAY – TUESDAY

In-depth half- and full-day Tutorials  
Multi-day Training Classes continue

## WEDNESDAY – THURSDAY

Keynotes, Concurrent Sessions, the EXPO, Networking Events,  
Receptions, Bonus Sessions, and more

## FRIDAY

Testing & Quality Leadership Summit



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

REGISTER BY JULY 30, 2010  
AND SAVE UP TO \$400.  
GROUPS OF 2 SAVE EVEN MORE!

 SOFTWARE  
QUALITY  
ENGINEERING  
**CONFERENCES**  
[www.sqe.com](http://www.sqe.com)



A photograph of two people climbing a dark, craggy rock face against a clear blue sky. One person, wearing a green shirt and dark shorts, is higher up and reaching down to help the other person. The second person, who is shirtless and wearing dark pants, is lower down and reaching up. The scene is backlit, creating a silhouette effect and a bright glow around the figures.

# Working *Together*— Not Just Working Together

by Johanna Rothman

**S**OPHIE strode down the hall to Randy's office. "Randy, what is this?" she asked as she waved a sheaf of papers.

Randy looked up and said, "I have no idea. What are you waving around?"

"Your plans for your group!"

"Oh, my predicted project portfolio for the next six months?"

"Yes! When were you going to talk to me? I'm the development manager! If you're not testing the systems we're developing, what the heck are you doing?"

"Well, this is what I've been trying to discuss with you for the past few weeks, but you haven't made time to see me. So, I thought I'd publish my portfolio and get your attention. It worked, eh?"

Sophie and Randy are working together, but they are not collaborating.

Collaboration—the ability to work jointly with others—is something we talk about a lot. We demand it in job descriptions. We assume the people who run the business do it. We expect it of teams. But, how do people really collaborate?

Assuming you can get people to sit down together, a good way to think about collaboration is to think first about the steps to collaboration.

## Barriers to Collaboration

In my experience, team size matters, too. If you have a small team—fewer than ten people—you have a good chance of making collaboration work. But, if you are trying to get ten or more people together to collaborate, it's much harder.

That's because it's just about impossible to create a team of ten or more people. Groups of more than nine people splinter into smaller groups of three to seven people.

In addition to team size, there are several barriers to collaboration that organizations impose—perhaps unintentionally—on people:

**Playing a zero-sum game:** A zero-sum game is a game where one person wins everything while another person loses everything. When you rank projects, there is only one No. 1 project. In a real sense, one project wins and the others lose. That's the only zero-sum game you want to play in an organization.

Otherwise, you are not optimizing the work at the highest level. If people can't decide which projects are ranked first, second, and third, they are preventing the organization from making progress. In that case, your competitors win.

**Hiding information:** Sometimes, people think that if they keep information hidden from others, they can enhance their position in the organization. And, for a short time, they may be able to do so.

**"If people can't decide which projects are ranked first, second, and third, they are preventing the organization from making progress."**

Solomon's steps	Why this step works
Deliver what you promise to deliver.	If you can't keep your promises, you are not trustworthy.
Be consistent in your actions and reactions.	People who act and react in inconsistent ways are not trustworthy.
Make integrity a cornerstone of your work.	Without integrity, can there be trust?
Be willing to discuss, influence, and negotiate.	Avoid becoming stuck on a position.
Trust in yourself and your colleagues.	Extending trust is a great way to earn it.

Table 1

## Steps to Collaboration

In *Building Trust in Business, Politics, Relationships, and Life*, Robert C. Solomon claims there are five steps to building a trusting relationship, as shown in table 1.

Notice that the last step is extending trust. Sophie is surprised by Randy's plans. He is promising to deliver some work—just not the work she wants done. But, Sophie hasn't been consistent in her actions and reactions, because she hasn't made time to work with him. If she refuses to make time to work on the portfolio with Randy, she hasn't been trustworthy.

But, in the long term, hiding information prevents people from collaborating. Remember, although you might be negotiating with a peer about a problem, it's in the organization's best interests to have all the information out and available.

**Incentives prevent teamwork:** I've worked with managers whose salespeople were paid when an order was booked, not when it was shipped. There was no incentive for salespeople to sell what the development team had already created. In fact, there was tremendous incentive to sell new things that hadn't been invented yet.

You've probably met project managers who had management by objectives (MBO) that only dealt with the project's release date. Those project managers worked with development managers whose MBO were about features and with test managers whose MBO were about defects. They have no incentive to collaborate in this situation. In fact, it is not in anyone's interest to collaborate. Except for the poor customer who desperately wants them to collaborate.

**Decisions don't stick:** It's even worse when a group of people get together, make a decision (or several decisions), and then someone with more positional power changes that decision. Who wants to bother collaborating in that situation? Any decision you make is as likely to get thrown out as stick.

**The time-space continuum prevents true collaboration:** Many of you work in geographically distributed teams. You know how hard it is to collaborate on product development when you are not in the same place. It's even more difficult for managers to collaborate on decisions when they are not in the same place.

One CIO had five remote teams run by directors: two in the US, one in the UK, one in Poland, and one in Bangalore. Each director had his own MBO, which was not in concert with the others, and they never met as a group. Some subset of directors would get together and make a decision; the next day, when the others met, they made a different decision.

The CIO had finally had enough. "I want you folks to work together on this!" he mandated. Once he called the meeting, the directors were finally able to air their concerns and explain why they'd made their original decisions. Collaboration cannot occur without interaction among all the players at one time.

## Rewind Randy and Sophie

Once Sophie realized she'd ignored Randy, she decided to make the effort to collaborate on the project portfolio.

They picked a mutually convenient time to meet. They each considered the principles behind their decision making. Luckily, they did have a common goal. They discussed how to make the entire organization successful with their planned portfolio, and they agreed on when to review the portfolio again.

As Sophie and Randy worked through their plans for the next quarter, they kept reminding each other about pending deliverables and showing each other their progress. When one of Sophie's projects encountered trouble, she explained the problem to Randy. They were able to make other plans and manage the problem.

Once they realized the power of collaboration, Sophie and Randy brought in others across the organization, not only to manage the project portfolio but also to collaborate on product roadmaps and do a little strategic planning.

It doesn't matter where you are in the organization; collaboration is necessary. If you're a tester, you need to work with developers and vice versa. If you're a product owner, you need to work with your users. If you're a manager, you need to work with your teams. The higher the stakes, the more collaboration will help you. **(end)**

jr@jrothman.com

**Sticky Notes**

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

■ References

## Real Collaboration Can Produce Team Flow

When people collaborate, you can see the team members get into flow. I once worked on a large project where we all had the same vision of the product; we all worked together, delivering small, working chunks of code into the code base; we had demos for one another; and we treated one another with respect and had a great time.

We were in team flow. Because we had small, frequent deliverables, we could see how each person delivered on a daily or almost-daily basis. Because we had the same vision, we were able to be consistent in our actions and reactions. Any time anyone outside the team asked us to cut corners (this is where integrity is key), we had our vision against which to check our work. And, even though we had plenty of disagreements, we stuck to a principle, rather than to a position. We delivered a new software-hardware combination product in nine months—a record for the organization.

Watch out for the inadvertent organizational barriers to collaboration. And don't just work together—try real collaboration.



## BambooRM

RESTON, VA—Bamboo Solutions has announced that it is launching BambooRM, a SaaS-based requirements management program that requires no installation, is easy to use, connects workplaces all over the globe, and is available via monthly subscription. BambooRM allows project managers and teams to collaborate on projects more easily with one repository of project requirements.

BambooRM simplifies requirements management by integrating use cases and business and functional requirements all in one place. Managers, business analysts, consultants, and team leaders can view, manage, and trace project requirements online in their browsers, including:

- Creating business requirements for any product or service
- Defining and associating sub-level business requirements
- Associating use cases with multiple associated business requirements
- Associating functional requirements with each business requirement
- Prioritizing business and functional requirements
- Tracing the project requirements
- Creating and printing requirements with easy-to-read summaries

For additional information, visit [bamboosolutions.com](http://bamboosolutions.com).

## End-to-End Application Lifecycle Management Solution

CINCINNATI, OH—Seapine Software has released its ALM solution, which streamlines development and QA processes to improve software quality, increase productivity, and lower development and business costs.

Seapine's ALM solution includes TestTrack RM for requirements management, TestTrack TCM for test case management, TestTrack Pro for issue and defect management, Surround SCM for software configuration management, and QA Wizard Pro for automated functional and regression testing. TestTrack RM, the newest addition to the Seapine ALM product family, enables development teams to manage all aspects of requirements including planning, traceability, impact, review processes, measurement, and reporting.

Seapine's ALM solution includes scalable, feature-rich, team-based tools that can be used separately to manage each critical phase of the development lifecycle or seamlessly integrated to ensure quality at every stage of development. These tools are cross-platform and built on flexible architectures using open standards to support today's popular development methodologies and industry best practices. Seapine's ALM solution gives today's organizations the collaboration, workflow automation, reporting, security, and extensibility capabilities to streamline communications, improve traceability, and make development and QA teams more productive.

For additional information, visit [seapine.com/products.html](http://seapine.com/products.html).

## Concerity Analytics

SALT LAKE CITY, UT—Concerity has announced the free beta launch of Concerity Analytics, a new type of integrated toolkit designed specifically to help software companies measure real-world usage of their software applications.

Designed initially for Windows desktop and ASP.NET software products, Concerity Analytics allows independent software vendors to gather key data on how their software products are used in the field by end users—either in a test environment or in production.

Concerity Analytics provides software companies and in-house software teams with the ability to:

- Measure and optimize feature usage
- Understand beta tester behavior
- Track, understand, and improve product quality
- Prioritize product development efforts
- Track and validate marketing programs (including conversion strategies for “trialware”)
- Proactively determine where support needs are highest and thus lower support costs
- Increase engineering return on investment
- Outmaneuver the competition through faster, smarter, and more effective use of resources

For more information, visit [concerity.com](http://concerity.com).

## Coverity 5

SAN FRANCISCO, CA—Coverity, Inc. has unveiled Coverity 5, the industry's first software integrity offer that automatically scans, prioritizes, and maps the impact of defects introduced by software changes. This new offer is specifically designed to help development organizations mitigate the business risk of software changes across their entire product portfolio.

The new Coverity 5 defect impact mapping capability is the first to enable developers to automatically map and identify how a single defect impacts multiple code bases, projects, and products across the portfolio.

The new Coverity 5 unified defect management interface is the first to enable developers and management to review, prioritize, and triage their C/C++, Java, and C# defects in a single workflow, providing a single version of the truth for the state of integrity across the entire product portfolio.

Developer productivity enhancements:

- New, state-of-the-art code browser provides advanced defect drill-down capabilities, easy-to-use defect markers, shared views, and in-line expansion into inter-procedural defect details.
- Advanced defect reporting provides an easy way to track fixed defects, defect trends, the overall state of integrity across the entire product line, and evidence for defect remediation for compliance reporting.
- Robust scalability enhancements enable more concurrent users and simultaneous analysis commits.

For more information, visit [coverity.com](http://coverity.com).

### MINGLE 3.0

SAN FRANCISCO, CA—The ThoughtWorks Studios team has announced that Mingle 3.0 is now available for download. The new features and enhancements in 3.0 continue to close the collaboration and program management gap that exists between business stakeholders, project managers, business analysts, QA, and development engineers.

Features include:

- New team collaboration features using Murmurs
- Powerful program management capabilities
- Usability and support enhancements in the Macro Editors that extend reporting features
- Clustering for larger deployments
- Support for Subversion 1.6.5
- Other features to enable users to view and access development data in new ways

For more information, visit [thoughtworks.com/studios](http://thoughtworks.com/studios).

### ALM SOFTWARE

LAFAYETTE, CA—TechExcel, Inc. has announced it is offering software developers a free, ten-user license for DevSuite, its award-winning suite of ALM tools. The new 10-Users Free Program gives small development teams an excellent, no-risk opportunity to experience how the company's fully integrated set of enterprise-class tools can help them more efficiently and effectively manage all phases of application development.

DevSuite is a fully-integrated ALM system consisting of TechExcel's popular DevTrack for defect and project tracking; DevSpec for requirements management; DevPlan for project planning; and KnowledgeWise, a wiki-based knowledge management system. Available as an integrated client/server and Intranet/Internet solution, DevSuite provides powerful workflow and process automation, robust searching and reporting, and simple point-and-click customization. It also features an intuitive user interface, universal ODBC support, advanced email notification, built-in time tracking, extensive customization options, and presentation-quality reports and graphics. DevSuite also enables software development organizations, regardless of size, to manage globally-distributed development teams efficiently and effectively.

Managers interested in participating in the program can register and download DevSuite to begin their thirty-day evaluation and receive a free permanent license for up to ten team members.

For additional information, visit [techexcel.com/products/devsuite/devsuite.html](http://techexcel.com/products/devsuite/devsuite.html).

### Agile Platform 5.0

SAN RAMON, CA—OutSystems has released the new Agile Platform 5.0. Agile Platform integrates the two lifecycles of application development and business process management within a single, powerful environment—enabling the automation of business processes and delivery of flexible Web applications that are built for continuous change.

With this version, OutSystems extends the Agile Platform's existing capabilities for building rich Internet applications by adding a new business process technology layer, enabling IT organizations to model, integrate, deploy, execute, monitor, and optimize human-to-system processes that are 100 percent in sync with the applications that support them.

The new EPA technology allows simplified deployment of process definitions with innovative support for automating human-to-system interactions. A new TaskBox feature allows organizations to streamline the deployment of new and changed business processes with automatic task notification, intelligent workflow support, and process documentation at users' fingertips.

Agile Platform 5.0 features include:

- Supporting in-flight validation
- Copy and paste across a wide range of model objects
- Improved support for Web service integration
- Simplified support for the delivery of multilingual applications

For additional information, visit [outsystems.com](http://outsystems.com).

### Parasoft C++test

MONROVIA, CA—Parasoft Corporation has announced the release of Parasoft C++test, an integrated solution for automating a broad range of best practices proven to increase software development team productivity and software quality for C and C++. In this new release, Parasoft significantly expands the scope of C++test with the introduction of integrated runtime memory analysis that is suitable for both enterprise and embedded development.

New capabilities for reducing the risks of C and C++ software development:

- Runtime memory analysis enables teams to automatically identify serious runtime defects—such as memory leaks, null pointers, uninitialized memory, and buffer overflows—during execution.
- Parasoft Concerto Task Assistant brings change-based testing and automated task management directly to the developers' familiar work environment.
- Extended environment and compiler support for ARM, Keil, IAR, Texas Instruments, QNX, AIX, and Eclipse 3.5 enables more developers to use Parasoft C++test as a seamless part of their development process.

To learn more, visit [www.parasoft.com/c\\_cpp\\_testing](http://www.parasoft.com/c_cpp_testing).

### TestRail 1.0

Gurock Software has announced the immediate availability of its new test management solution TestRail 1.0. TestRail is a comprehensive, Web-based test case management application to manage, track, and organize software testing efforts.

TestRail provides built-in solutions for managing projects, milestones, and releases, as well as test configurations and advanced testing scenarios.

Productivity- and team-related features such as personalized todo lists, email notifications, and team conversations are a core concept of TestRail and integrate seamlessly with the daily workflow. The built-in reporting functionality helps teams get real-time insights into their testing process and progress.

For more information, visit [www.gurock.com/testrail/](http://www.gurock.com/testrail/).

### LISA 5.0

DALLAS, TX—iTKO has announced the general availability of its flagship LISA product suite version 5.0.

LISA 5.0 features include:

- Powerful and versatile virtual service environments. LISA captures and models inaccessible or unavailable IT resources into virtual service environments that produce the same dynamic behavior, performance, and data as the real production systems.
- Improved end-to-end transparency, automation, and validation. LISA's Pathfinder provides complete transparency to trace errors to their source across complex architectures spanning multiple application tiers and heterogeneous technologies.
- New Pathfinder Pro streamlines defect collaboration with no new skills required. LISA's new Pathfinder Pro builds upon Pathfinder's transparency and traceability capabilities by enabling fast, streamlined resolution of defects without any new technical skills or processes required for development and quality assurance teams.

For more information, visit [www.itko.com/lisa](http://www.itko.com/lisa).

### ElectricCommander 3.5

SUNNYVALE, CA—Electric Cloud has released ElectricCommander 3.5, a fully customizable and extensible version of its tool for automating and managing the build-test-deploy process in software development. Electric Cloud also announced significant upgrades to ElectricAccelerator ([www.electric-cloud.com/news/2010-0216a.php](http://www.electric-cloud.com/news/2010-0216a.php)).

The customizability of ElectricCommander 3.5 simplifies software production management and removes the need for developers to learn and manage multiple tools.

ElectricCommander automates and manages the error-prone, manual pieces of the build-test-deploy process, making software production faster and more efficient. ElectricCommander 3.5, available immediately, integrates seamlessly into a development team's current environment with fully customizable user interfaces, including dashboards and forms. Version 3.5 also provides the tools to create and share custom plug-ins for smoother integration with application lifecycle management tools, homegrown tools, custom workflows, etc.

For more information, visit [www.electric-cloud.com](http://www.electric-cloud.com).

Have news you'd like to share?

Send your new product announcement to

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

## Your Task: Drive the Nail



## Select a Tool:



Choosing a tool is easy when you know what the job requires.

BugHost has been providing affordable solutions for both **Enterprise and Small Business** for nearly 10 years. Start your free 30-day trial today to see why BugHost is the right tool for you.



- ✓ Bug Tracking
- ✓ Requirements
- ✓ Test Cases
- ✓ Task Tracking
- ✓ Free Technical Support

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

**BugHost™**

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

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



# FAQ

expert answers to  
frequently asked  
questions

by Claire Lohr  
clohr@computer.org

## How can I ensure my requirements are fully testable?

There is not just one answer to this question. Testability, like beauty, is in the eye of the beholder. It all comes down to successful communication between the requirements initiator and the tester. There is not one universal answer, because every pair of requirements initiator and tester is different. The challenges facing testers can range from very high integrity systems such as NASA on one end, down to minimal-documentation agile projects at the other end, and everything in between. All address the testability issue with different solutions.

NASA has answered this question by developing the Automated Requirements Measurement Tool ([satc.gsfc.nasa.gov/tools/download/](http://satc.gsfc.nasa.gov/tools/download/)). Traditional requirements documentation can be developed following the *Std 830 IEEE Recommended Practice for Software Requirements Specifications*. Use cases can be developed using internal or UML standards. The agile community starts with user stories (described by the Agile Manifesto). While these are the most popular requirements documentation techniques, there are many more. Each type of requirements standard enhances the testability of the system by providing the tester with more complete and correct information—the standards are developed by consensus and contain a compendium of useful information from many experienced practitioners.

The following techniques can facilitate the testing process within any documentation or tool format:

- Include quantification in each requirement—use actual numbers or the words “any” or “none.” Avoid vague terms such as “user friendly.”
- Include visual representations—many humans are visual learners; they understand pictures better than words. Techniques such as the decision table, state transition diagrams, and the model in use cases all provide visual representations.
- Expect to develop requirements iteratively. Do the iteration then have the peer reviews (walk-throughs or inspections) and expect updates throughout development.
- Collect metrics to show the rewards of early requirements correction and the cost of neglecting the requirements definition phase of the lifecycle.
- Build a team with a variety of skills (e.g., testing methodology, business analysis, tools gurus) and keep team member morale as high as possible.
- Have online examples with blank templates and “good” examples from real projects for new team members to use as models.
- Use more than one technique.
- Get both your testers and business analysts certified, as the pre-exam preparation will cover the most important requirements and testing techniques and issues.

Crafting testable requirements is part art and part science. There are a lot of helpful tools and techniques available to guide your requirement creation, helping to ensure a high level of clarity and testability.

# The Unshreddable Résumé

These days, competition is stiff for the few job openings available. These five tips can help keep your résumé out of the trash and get your foot in the door.

by **Heather Shanholtzer** | [hshanholtzer@sqe.com](mailto:hshanholtzer@sqe.com)

Even if you weren't one of the millions of people faced with layoffs this past year, it's always a good idea to keep your résumé up to date. You never know when opportunity may come knocking or you'll find yourself unexpectedly "exploring a different avenue."

Your résumé can make or break you as a candidate for a job. So, it's wise to make the time to put together thorough, organized documentation of your skills, accomplishments, and experiences.

A résumé is a marketing tool, but it's important to remember that it's not all about you. It's an opportunity to show a potential employer that you are the one person who has the skills needed to fill the open position and add value to her organization.

Two important things you can do to increase your odds of landing an interview are: make sure your résumé explicitly defines your experience, be it as a tester, developer, manager, or business analyst; and tailor your résumé specifically to the job you seek. Potential employers want to know at a glance that you have the skills needed to do the job. If a hiring manager has to wade through ten pages of unrelated work history to find out whether you have Java experience, chances are good your résumé will end up in the trash.

## Five Tips for Attention-grabbing Résumés

So, what can you do to escape the dreaded shredder and get some face time with the hiring team? These five things can help you set your résumé apart.

**Tell them what they need to know:** The most effective résumés focus on a specific job title and address succinctly the employer's stated job requirements. Michael Kahn, a software professional with nearly twenty years of experience and author of *The Software Hiring Handbook: The Software Developer's Guide to Conducting a Job Interview*, says he likes to see evidence of an applicant's experience level with a certain skill—rather than a vague buzzword—explicitly addressed in a résumé. "If I need someone to maintain and develop an XML parser, then having a statement on a résumé such as 'interfaced with an XML parsing engine' or 'worked with configurations stored in XML format' is more meaningful than merely listing 'XML' in a laundry list of keywords of 'skills.'"

**Highlight your accomplishments:** Many people are uncomfortable patting themselves on the back. But, your résumé

is one place that it's actually necessary for you to expound on the great work you've done. If you managed a project that came in \$2 million under budget as a result of your leadership, say so. In Michael's experience, identifying tangible results, such as "reduced defects by 10 percent," or otherwise defining accomplishments that provided an organization time or cost savings is a good way to make a positive impact on the hiring manager.

**"A résumé is a marketing tool, but it's important to remember that it's not all about you."**

Michael also has an eye for detail. "Rather than just saying 'implemented driver code in C,'" he says, "I am more interested in a candidate that tells a bit more of the story, such as 'implemented IEEE-1394 driver in C, and participated in unit test and integration of the driver with the product lineup.'"

**Proofread before you send:** Typos and mistakes aren't necessarily a deal breaker, but they can affect your chances of landing the job you want. "Too many grammatical errors or sloppy mistakes will likely cause the résumé to head toward the discard pile," Michael says. "While I won't discard a résumé automatically because of one or two errors, if it appears that the résumé was not read through prior to submitting, then it likely would be discarded."

While a few mistakes may not disqualify you for some positions, a typo-riddled résumé can affect the perception of your ability to perform well in certain jobs. For example, a tester who hasn't taken the time to run spell check is not likely to be the top contender for a QA position, as the hiring group may feel that quality is not her highest priority.

**Keep it short and sweet:** There are a lot of people looking for jobs these days and not a lot of companies that are hiring. So, when a position does open the manager is going to be swamped with résumés. The volume of applicants normally precludes a manager from spending more than a few seconds scanning your résumé for applicable skills. If those skills are not highly visible, your résumé likely is headed to the trash.

"In the age of word processors, it is fairly easy for a candidate to present a tailored résumé for a job description that should emphasize their most relevant skills and experiences for that position," Michael says. "To me, it feels like a lack of effort to receive a nine-plus-page résumé with lots of generic experience for a specific job position."

"The reality of a job search is that candidates should make their relevant experience 'jump out' at the reviewers, and not expect the reviewer to scrutinize a long résumé to extract de-

tails that are buried on page four or beyond.”

**Show you play well with others:** With more organizations adopting agile methodologies and the increase in globally distributed teams, it has never been more important for you to demonstrate your ability to work closely with others. When reviewing your résumé, a hiring manager will be looking for evidence that you are a team player. “In today’s workplace,” Michael says, “a majority of software positions require team skills, so any statements that indicate team involvement (e.g., leading a team or interfacing with multi-discipline teams such as hardware, QA, etc.) are generally viewed as positive.”

## Tailor Your Résumé to Your Job Function

In addition to the ideas discussed above, what specific information should a tester, developer, manager, or business analyst include on her résumé that will show a hiring manager that this applicant is the right choice for the position? Johanna Rothman, an accomplished management consultant and author of several books, including *Hiring the Best Knowledge Workers, Techies & Nerds*, suggests an applicant consider the following questions when preparing her résumé. Answering these questions will help the hiring team determine your potential value to the organization.

**Testers:** What new or innovative approaches have you used

that helped the test effort discover more information?

**Developers:** How have you made changes that fit the product? How do you know a design is good?

**Software Managers:** How did you benefit your previous organization? How did you help people do their best?

**Business Analysts:** What problems did your work solve?

## Further Steps to Success

Whether you are recently laid off or you hope to land a great new job when the economy turns around, a well-written résumé is going to be a huge factor in whether you get an interview, but it’s not the only tool you need. Michael has a couple of suggestions for increasing your chances for a successful job search. “First,” he says, “nurture and maintain your professional network. In this industry, as with others, many people know each other, and your network can help you find job openings before they are posted.”

Michael also recommends staying abreast of new technology while you are looking for work. “With the advent of the Internet, and an open source project for just about anything,” he says, “it should be possible to devote a little time to most any area of software and learn more about it. Learning a new scripting language is usually a good place to begin. For an out-of-work job seeker, this ‘independent study’ is time well spent.” {end}

## index to advertisers

ADP West 2010	www.sqe.com/adpwest	24
AutomatedQA	www.automatedqa.com	19
Avnet	www.avnet.com	31
Better Software Conference	www.sqe.com/BSC	12
Better Software magazine	www.bettersoftware.com	23
BugHost	www.BugHost.com	41
Codenomicon	www.codenomicon.com	22
Hewlett-Packard	www.hp.com/go/agile	Back Cover
McCabe	www.mccabe.com	15
Microsoft	www.microsoft.com	Inside Front Cover
PureCM	www.purecm.com	18
Rally Software	www.rallydev.com/bsm	Inside Back Cover
Ranorex	www.ranorex.com	9
Seapine	www.seapine.com	1
SQE Training—eLearning	www.sqetraining.com/eLearning	16
SQE Training—Live Virtual	www.sqetraining.com/VirtualTraining	6
STARWEST 2010	www.sqe.com/STARWEST	35
TechExcel	www.techexcel.com	2
ThoughtWorks	www.thoughtworks.com	5
Web Performance	www.webperformanceinc.com	10

**Display Advertising**  
advertisingsales@sqe.com

**All Other Inquiries**  
info@bettersoftware.com

*Better Software* (USPS: 019-578, ISSN: 1553-1929) is published six times per year January/February, March/April, May/June, July/August, September/October, November/December. Subscription rate is US \$40.00 per year. A US \$35 shipping charge is incurred for all non-US addresses. Payments to Software Quality Engineering must be made in US funds drawn from a US bank. For more information, contact info@bettersoftware.com or call 800.450.7854. Back issues may be purchased for \$15 per issue (plus shipping). Volume discounts available. Entire contents © 2010 by Software Quality Engineering (330 Corporate Way, Suite 300, Orange Park, FL 32073), unless otherwise noted on specific articles. The opinions expressed within the articles and contents herein do not necessarily express those of the publisher (Software Quality Engineering). All rights reserved. No material in this publication may be reproduced in any form without permission. Reprints of individual articles available. Call for details. Periodicals Postage paid in Orange Park, FL, and other mailing offices. POSTMASTER: Send address changes to Better Software, 330 Corporate Way, Suite 300, Orange Park, FL 32073, info@bettersoftware.com.



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

**Erik Huddleston**  
CTO  
Inovis



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

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

# HARNES

the power of Agile.

## HP Application Lifecycle Management solutions

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

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

Outcomes that matter.

Get more information at  
[hp.com/go/agile](http://hp.com/go/agile)

