



# The Award-Winning-Est Agile Lifecycle Management Solution



Three-time Jolt Product Excellence award winner in 2006, 2007 and 2008 for project management tools



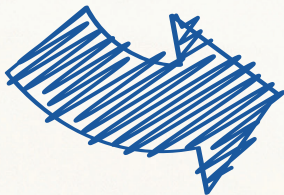
QSM concludes—

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



Forrester Research says—

"Rally designed the requirements management capabilities in Rally Enterprise, a software-as-a-service (SaaS) ALM solution, to suit teams using Agile processes. The product performs flawlessly in this regard..."



Get your FREE trial of Rally Enterprise at

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

No download, no installation, no commitment



Visit us at the Agile Development Practices Conference, booth #200  
November 10 - 14, Orlando, FL

# BETTER SOFTWARE

Dear *Better Software* magazine subscriber,

As an attendee of a Software Quality Engineering event, you receive a one-year digital subscription to *Better Software* magazine included in your paid conference registration. Please take a moment to confirm that you would like to receive/continue to receive *Better Software* magazine?

[CLICK HERE](#)

Thank you for your participation. I look forward to seeing you at future Software Quality Engineering conferences.

Sincerely,  
Lee Copeland  
Conference Chair &  
Managing Technical Editor,  
*Better Software* magazine



# BETTER SOFTWARE

Name

Email

Birth Month

Do you wish to  
continue receiving the  
*Better Software* ezine?



Which SQE event  
did you attend?

SUBMIT





November 2008

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

# BETTER SOFTWARE

**SIMPLIFY, SIMPLIFY**  
Status reports  
executives can use

**UNCHARTED  
TERRITORY**  
The route to great  
test coverage

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

*GETTING*  
**AGILE** with  
USER-CENTERED  
DESIGN  
*CREATE SOFTWARE  
USERS CAN'T LIVE WITHOUT*

**FIND THE BUG INSIDE & WIN  
AN AMAZON GIFT CARD!**



# How much effort does it take to determine the status of your agile projects?

With the VersionOne agile project planning and management system, the answer is: almost none. Our centralized platform consolidates up-to-date project data to give you comprehensive project visibility at the click of a button. Not only will your team save time with our built-in agile process workflow, they'll also experience enhanced collaboration and higher productivity.



VersionOne.  
More visibility.  
Less overhead.

Let us help you simplify and accelerate your agile software development.

For a free 30-day trial, go to <http://www.versionone.com/bettersoftware>.

# Looking for a better way to manage test cases?



Learn how to write and manage test cases more effectively. Read our TestTrack TCM best practices white paper.

Download it today at [www.seapine.com/bstcmbp](http://www.seapine.com/bstcmbp)



Seapine

## TestTrack® TCM

Software for test case planning, execution, and tracking

You can't ship with confidence if you don't have the tools in place to document, repeat, and quantify your testing effort. TestTrack TCM can help you thoroughly test your applications in less time.

In TestTrack TCM you have the tool you need to write and manage thousands of test cases, select sets of tests to run against builds, and process the pass/fail results using your development workflow.

With TestTrack TCM driving your QA process, you'll know what has been tested, what hasn't, and how much effort remains to ship a quality product. Deliver with the confidence only achieved with a well-planned testing effort.

- Ensure all steps are executed, and in the same order, for more consistent testing.
- Know instantly which test cases have been executed, what your coverage is, and how much testing remains.
- Track test case execution times to calculate how much time is required to test your applications.
- Streamline the QA > Fix > Re-test cycle by pushing test failures immediately into the defect management workflow.
- Achieve complete traceability between test cases and defects with seamless TestTrack Pro integration.

### Seapine ALM Solutions:



**TestTrack Pro**  
Issue & Defect Management



**TestTrack TCM**  
Test Case Planning & Tracking



**Surround SCM**  
Configuration Management



**QA Wizard Pro**  
Automated Functional Testing

 Seapine Software

Download and evaluate TestTrack TCM today at  
[www.seapine.com/bstcm08](http://www.seapine.com/bstcm08)



# SOFTWARE TESTING ANALYSIS & REVIEW

*The Greatest Software Testing Conference on Earth*



**MAY 4-8, 2009**

**The Rosen Centre Hotel**

*Concurrent Sessions Covering:*

Test Management  
Test Techniques  
Test Automation

Agile Testing  
Test Processes  
Exploratory Testing

Metrics  
Outsourcing  
Performance Testing

Security Testing  
Web Services

**Over 98% of 2008  
Attendees Recommend  
STAREAST to Others  
in the Industry**

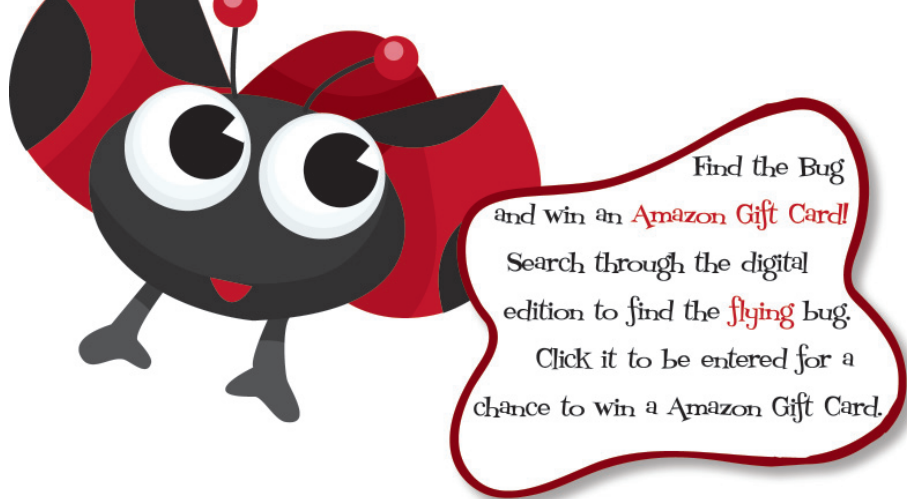


**[www.sqe.com/stareast](http://www.sqe.com/stareast)**

**REGISTER EARLY AND SAVE \$200!**







## Cover Story

### GETTING AGILE WITH USER-CENTERED DESIGN 22

Agile practices go a long way toward providing value to our customers. But in today's market, we must endeavor to adopt a more user-centered approach to create products our customers can't live without. *by Jon Dickinson and Darius Kumana*

## Features



## Columns & Departments

### In Every Issue

Mark Your Calendar 4

Contributors 6

eLightenment 8

Product Announcements 42

10 Things You Might  
Not Know About ... 46

Ad Index 48

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

**Subscribe today to get ten issues.**

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

### GOOGLE WEB TOOLKIT 26

In part two of the series, Daniel introduces Google Web Toolkit's testing infrastructure and demonstrates how to build an Ajax application test first.

*by Daniel Wellman*

### SIMPLE SUMMARIES OF COMPLEX PROJECTS 34

How can we meaningfully summarize—in a brief status report without losing important details—the successes and setbacks our projects experience? *by Payson Hall*

### TECHNICALLY SPEAKING 13

**Follow the Process** • *by Lee Copeland*

Building better software does not rely on methodologies. "Following the process" omits important human factors that ultimately lead us to success.

### CODE CRAFT 14

**Encapsulation and Vampires** • *by Kevlin Henney*

Encapsulation is more than just using the "private" keyword when defining a class. You need a boundary that keeps the vampires out.

### TEST CONNECTION 18

**Cover or Discover?** • *by Michael Bolton*

Excellent testing isn't just about covering the "map"—it's also about exploring the territory, which is the process by which we discover things that the map doesn't cover.

### MANAGEMENT CHRONICLES 20

**Keep Non-developers in the Loop** • *by Melanie Tayler*

Keeping QA members up to date on changes as they happen—through meetings, wikis, and email—can reduce the number of unnecessary bug reports and save you time and frustration.

### THE LAST WORD 47

**Metrics that Motivate** • *by Linda Hayes*

To implement a meaningful incentive system for your team, you need to select metrics that encourage the behaviors you need and the results you want. But first you have to decide *what* you need and want.

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

## MARK YOUR CALENDAR

### TRAINING WEEKS

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

#### Testing

**November 17–21, 2008**

Tampa, FL

**March 23–27, 2009**

Boston, MA

#### Agile Software Development

**March 30–April 3, 2009**

Washington, DC

**April 20–24, 2009**

San Francisco, CA

### SOFTWARE TESTING CERTIFICATION

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

**November 18–20, 2008**

Sunnyvale, CA

**December 2–4, 2008**

Phoenix, AZ

**February 17–19, 2009**

Houston, TX and Toronto, ON

### CONFERENCES

#### STAREAST 2009

##### Software Testing Analysis & Review

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

**May 4–8, 2009**

The Rosen Centre Hotel

Orlando, FL

#### Better Software Conference & EXPO 2009

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

**June 8–11, 2009**

The Venetian

Las Vegas, NV

# BETTER SOFTWARE

Publisher

**Wayne Middleton**

Vice President of Publishing

**Holly N. Bourquin**

Editor-in-Chief

**Heather Shanholtzer**

#### Editorial

Managing Technical Editor

**Lee Copeland**

Technical Editors

**Chuck Allison**

**Jonathan Kohl**

**Antony Marciano**

Editor, StickyMinds.com

**Francesca Matteu**

Managing Editor, Multimedia

**Joseph McAllister**

Production Coordinator

**Cheryl M. Burke**

#### Design

Creative Director

**Catherine J. Clinger**

#### Advertising

Senior Advertising Sales Manager

**Shae Young**

Advertising Sales Manager

**Joe Anderson**

Production Coordinator

**April Evans**

#### Circulation and Marketing

Marketing Coordinator

**Sidney White**

Circulation Coordinator

**Jamie Green-Gago**

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





Take quality to the next level



# DevTest Studio

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

## DevTrack

Use DevTrack to track defects/issues

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

## DevTest

Use DevTest to manage your testing

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

## TestLink

Use TestLink to automate your testing

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

**TechExcel**

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





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



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



**PAYSON HALL** is a consulting systems engineer and project manager from Catalysis Group, Inc. in Sacramento, CA. Formally trained as a software engineer, Payson has performed and consulted on a variety of hardware and software systems integration projects in both the public and private sectors throughout North America and Europe during his twenty-six-year professional career. He is a regular columnist on StickyMinds.com.



**LINDA G. HAYES** is a founder of Worksoft, Inc., developer of next-generation test automation solutions. Linda is a frequent industry speaker and award-winning author on software quality. She has been named one of *Fortune* magazine's People to Watch and one of the Top 40 Under 40 by *Dallas Business Journal*. Linda is a regular columnist and contributor to StickyMinds.com and *Better Software* magazine, as well as a columnist for *Computerworld* and *Datamation*, author of *The Automated Testing Handbook*, and co-editor with Alka Jarvis of *Dare to be Excellent*. You can contact Linda at [linda@worksoft.com](mailto:linda@worksoft.com).



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



**CLAIRE LOHR** has been an active professional in the computer field for forty years, with the past twenty years focused on software process improvement. Claire currently provides training and consulting services for a wide variety of both government and commercial clients. She is a Lloyd's Register-trained ISO 9000 Lead Auditor. Claire has been trained to perform software capability evaluations for the SW-CMM. She chaired the IEEE 829-2008 Working Group, and has served on both the IEEE Computer Society's Software and Systems Engineering Standards Committee (S2ESC) and the IEEE Computer Society's Standards Advisory Board (SAB; as Vitality Chair).



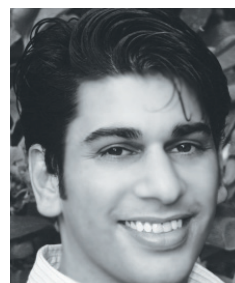
**MELANIE TAYLER** is the quality assurance project lead at Macadamian Technologies. She has spent her professional career developing and driving the company's quality processes while adapting them to the challenges of agile methodologies. You can reach Melanie at [melanie@macadamian.com](mailto:melanie@macadamian.com).



**DANIEL WELLMAN** is a technical lead at Cyrus Innovation ([www.cyrusinnovation.com](http://www.cyrusinnovation.com)), a leading New York agile consultancy where he leads development projects and coaches teams on adopting agile software development practices. Daniel has more than ten years of experience building software systems and is an expert in agile methodologies, object-oriented design, and test-driven development. Contact Daniel at [dan@danielwellman.com](mailto:dan@danielwellman.com).



**JON DICKINSON** is an independent software development consultant based in the UK. He specializes in using agile software development principles and techniques to help teams improve the consistency of their software delivery successes. Jon is currently co-authoring a training course to help teams integrate agile software development with user-centered design. He is also writing a book, which is due out at the end of spring 2009, on Web application development for the new Java/Groovy-based framework, Grails. Where Mars meets Venus, Jon remains focused on finding ways for developers and designers to jell into effective delivery teams.



As head of user experience at Isotrak Ltd., **DARIUS KUMANA** ensures that the company's agile product development approach remains focused on the goals, needs, and desires of real people. Previously, Darius was head of new media at Cambridge Assessment (University of Cambridge), where he was tasked with growing a center of excellence for user-centered design. Darius has worked across a variety of industries, including finance, education, retail, logistics, and nonprofit. His experience ranges from multinational companies to avant-garde firms and start ups. Where Venus meets Mars, Darius remains focused on successfully integrating UCD practices within agile environments. He is co-writing a new training course on "usability."

## EDITOR'S PICK

### Go Fish

In the past few weeks, I've received instant messages from strangers going by the names bracedtrout, reductivetroutrout, ornamental-trout, and nourishedtrout. At first, I thought I was just on someone's spam list. "Are you going to die?" one asked. "Let's discuss botany!" another said. I just ignored them and blocked the accounts from contacting me again.

But when I responded to one that asked, "What is your purpose in living?" by asking, "Friend or foe?" the trout on the other end asked, "Who is this?"

Aha! This trout doesn't know why I'm talking to him. The "trout" bit is just a middlefish, so to speak. Someone has set up a bot system on instant messaging software that prompts two strangers to converse over a random introductory thought. During another conversation, it became clear that I, too, was wearing the trout mask, when nourishedtrout asked me, "Are we close companions, Trout?" Hmm ... fishy.

I have to admit that it's an interesting concept, forcing two strangers into conversation. In fact, much of the Internet and its "social networking" are built on that very idea. But, sometimes, the hardest conversations to have can be between friends or coworkers—especially when communication or trust has slipped by the wayside.

In his October 2007 Management Chronicles, "Tending Communication Paths," Payson Hall writes about the importance of one-on-one communication, honesty, and openness between managers and employees. It's a great lesson in how to prevent your coworkers from becoming virtual strangers.

Read "Tending Communication Paths" at [www.StickyMinds.com/editorspick10-9](http://www.StickyMinds.com/editorspick10-9).

To read my troutbot chat transcripts and find out how to release any troutbots you've managed to hook, visit [www.StickyMinds.com/BlogTroutbot](http://www.StickyMinds.com/BlogTroutbot).



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

## Quotables

"When agile teams are given a problem to solve, you will see them literally swarm together as they begin to analyze and brainstorm solutions for the issue at hand. Like birds' flocking behaviors and the swarming of bees, this collective behavior is indicative of a complex adaptive system in which the group's intelligence is greater than the sum of its parts."

MICHELE SLIGER, "IS THIS CHAOS OK?"  
[www.stickyminds.com/quotables10-9a](http://www.stickyminds.com/quotables10-9a)

"But it's not just the people in the upper echelons who misjudge or underestimate how people will respond to change. When it comes to technological and organizational change, people in positions from team leader on up often seem to proceed on the assumption that everyone will march forward in lockstep. Employees one and all will appreciate the change, welcome it, and embrace it. Fair warning: This is unlikely to happen."

NAOMI KARTEN, "RECEPTIVENESS TO CHANGE"  
[www.stickyminds.com/quotables10-9b](http://www.stickyminds.com/quotables10-9b)

"Given the behavior changes—especially the enforced single tasking—you've now got a project that is far more likely to finish on time even though it's only 75 percent of the original duration. You can now do twelve months' worth of projects in nine months. That's a whopping 33 percent increase in capacity—for free. The biggest but least obvious benefit is that using this approach builds trust between managers and staff by removing all the unhealthy game playing that goes on with estimates."

CLARKE CHING, "CRITICAL CHAIN SCHEDULING FOR SOFTWARE PROJECTS"  
[www.stickyminds.com/quotables10-9c](http://www.stickyminds.com/quotables10-9c)



# ET

## RANDOM TESTING OR TECHNIQUE?

In the face of tight timelines and large, complex projects, do you need exploratory testing (ET) or is it just a waste of time?

In an ideal world, the phases of software development go hand-in-hand with extensive documentation. Specifications and requirements for each part are set out neatly, all ready for testers to begin work.

But that doesn't happen very often, does it? You'd want the tester to get involved as early as possible in the software development life cycle, so that development and testing can happen in parallel. But that's when the software is just beginning to take shape, is likely to be unstable and buggy, and will probably be accompanied with little or no documentation. And, change requests, increments, and iterations will be the order of the day; so even if you have documentation, it's likely to be outdated and of little use. Can testers do anything in such a scenario?

The answer lies in exploratory testing (ET). Using this technique, skilled testers can gain understanding of your product—they learn more about it, explore various areas to assess how they work, and get a feel of the risky areas that are likely to throw up bugs. This will help them design test cases, execute them, and use the findings in further test design and execution.

That makes it sound like ad hoc testing. But nothing could be further from the truth. Ad hoc testing is usually short-term; it consists of tests that are run only once, unless a bug is discovered. ET is a wider and deeper concept—it covers not only ad hoc testing, but all forms of testing. All testing, even scripted or automated testing, is exploratory to a certain degree, since tests could be modified on the

basis of results obtained and defects discovered. For instance, if a tester has run a particular test case and found unexpected results, the tester is bound to 'explore' further, and run a few more tests—which may not be part of the test suite—to discover more about the bug. That's exploratory testing to a certain extent.

### A Waste of Time?

ET is often considered a waste of time—though the tester spends a lot of time trying to understand how the software works and in which scenarios it is likely to fail; what you get at the end is a list of bugs, which in any case, you would also get with automated testing. ET would be a far more productive exercise if the testers could share their understanding of the product and suggest improvements based on that. That is possible if processes to capture this information are introduced.

Moreover, several challenges are associated with ET. Since test design and execution occur simultaneously, the test design cannot be reviewed in advance. This could lead to errors in the code and in the test cases.

Challenges also arise in documenting the test cases that testers create on the fly, especially if they need to be rerun in future. Since there is very little documentation, it is difficult to find out which test cases were run at which point of time. Thus, if the tests are run a second time, it may not be in exactly the same way. This can cause difficulties if it is important to document what parts of the product were functional at each point in time.

### Make Exploratory Testing Work

Most of these challenges will cease to be important if you have a skilled testing team; or if you outsource the testing function to a skilled test services provider. ET relies heavily on the testers' knowledge, experience, skills, creativity, and intuition. So, to make ET work for your project, it is important to find the right skills. To optimize your use of ET, well-defined procedures and documentation are required.

That's why QA InfoTech is the right choice for you, if you have software that needs to be tested rapidly, but has little supporting documentation. We have processes and metrics in place for weekend projects—those with little or no documentation, which need to be tested over the weekend; the results would be ready for you by Monday morning.

Our highly skilled test team formulates an assessment strategy, by understanding what to test, why it should be tested, how to test it, and when to stop testing. We then manage the testing process to yield the best results. Our approach includes effective and timely communication with project managers, developers, and the marketing team. We use the Staticator to facilitate communication regarding test efforts, coverage and so on.

We also have a well-defined methodical process to develop the test metrics or the test deliverables. Other documentation would include summary of defects; and how severe or otherwise these are, so that defect fixing can be prioritized.

## How Exploratory Testing Helps

ET helps in testing the product more holistically and offers scope for improvisation in the test design.

**Faster defect finding:** Since ET requires no pre-documentation or preparation, the bigger and more important defects are found faster than with scripted testing.

**Better scope for improvisation:** Since testers are not bound to run the entire scripted test suite, they can modify their test design and execution on the fly, if the situation and the context require it. This helps to speed up the testing process.

**More defect finding:** ET is essential for finding bugs that are not detected in the course of normal testing. There may be scenarios that

## Managing ET

### INDEX

#### STATICATOR: Indicating Status Made Simple

S. No.	Data	Description
1	Project Dashboard	
2	TestCase Execution Spreadsheet	
3	First Round Defects Spreadsheet	
4	Second Round Defects Spreadsheet	
5	Defects Summary by State	
6	Defects Summary by Severity	
7	Charts and Graphs	
8	Defect Containment Effectiveness and Defect Resolution Effectiveness	

## Managing ET

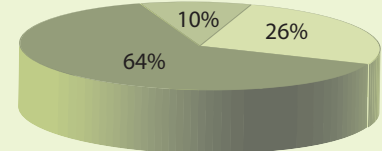
### Project QA Dashboard

Updated: [date] Last Build Tested: Project Build No. [xx], Name of Environment

Areas	Test Effort	Test Coverage (Approx.)	Quality Assessment	QAIT Comments
Login	Low	95%	Fine	
Functionality 1	Low	95%	Fine	
Functionality 2	Start	10%	In Process	
Functionality 3	Start	10%	In Process	
Functionality 4	High	5%	In Process	
Functionality 5	Start	5%	In Process	
Functionality 6	High	5%	In Process	
Functionality 7	Blocked	0%		
Functionality 8	Medium	15%	In Process	
Functionality 9	Low	95%	Fine	
Functionality 10	High	5%	In Process	
Functionality 11	Blocked	0%		
Functionality 12	Medium	40%	In Process	
Functionality 13	Medium	40%	In Process	

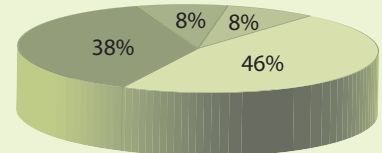
## Test Deliverables

### Defects by Priority



□ High □ Medium □ Low

### Defects by Severity



□ Major □ Minor □ Cosmetic □ Critical

the scripted tests haven't covered, but which could be covered to detect bugs through ET.

**Improving testers' productivity and creativity:** Through ET, testers learn new testing strategies, gain further skills and experience, and use their intellect and imagination to understand the product and testing requirements. This leads to increases in their productivity and skills for future projects.

**Better understanding of requirements:** ET provides an opportunity for testers to understand the product's requirements and functionality more thoroughly than if they were using any other testing methodology. The level of engagement of the tester with the product under ET is unparalleled.

So, if your software has to be tested on tight timelines, or you have incomplete requirements and documentation, contact us for rigorous, methodical, and in-depth ET. You can also use ET to verify the results of automated testing. ET can help you find defects that are less obvious, but critical, and have escaped detection in other forms of testing. •

For further information, visit [www.qainfotech.com](http://www.qainfotech.com).  
For business queries, please contact Mukesh Sharma at [mukesh@qainfotech.com](mailto:mukesh@qainfotech.com), or call +91-981-077-1714

## The Test Manager's Vade Mecum

BY FIONA CHARLES

Testers and test managers who come equipped with their own practices and tools can save time and effort and get a head start on their projects. Test manager and consultant Fiona Charles describes the "go with me" collection she has built over many years and projects to help leverage her varied experience and provide a quick start on new deliverables.

[www.StickyMinds.com/eLetterpick10-9a](http://www.StickyMinds.com/eLetterpick10-9a)

## Questions You Should Ask?

BY MICHELE SLIGER

It's a technique children and teenagers have mastered: asking "why?" until they get to an acceptable response (or until we're too tired to continue answering). Find out how Michele Sliger uses a similar approach designed by Six Sigma to drill down into the underlying cause of any problem within software projects. She then continues the inquisition with a series of additional questions in order to find out how these problems affect business value and technology. Learn what these questions are and how you can start using them to find out why things aren't going as planned.

[www.StickyMinds.com/eLetterpick10-9b](http://www.StickyMinds.com/eLetterpick10-9b)

## The Exceptional Exception

BY TOD GOLDING

So much more than a bucket for your errors, exceptions can be a valuable tool that lets you communicate to your clients not only that there is a problem but why and where the code failed.

[www.StickyMinds.com/eLetterpick10-9c](http://www.StickyMinds.com/eLetterpick10-9c)

## POWERPASS POINTER

### Mind the Gap

BY YURI CHERNAK

The requirements composition table is an effective technique comprising six steps that will help you assess an application's test coverage and identify gaps in your test suite even if you don't have any software requirements specifications.

[www.StickyMinds.com/eLetterpick10-9d](http://www.StickyMinds.com/eLetterpick10-9d)

A sampling of content from our eNewsletter archives

**StickyLetter:** August 6, 2008 [www.stickyminds.com/eLetter10-9](http://www.stickyminds.com/eLetter10-9)

## Hard Drive Heartache

by Holly Bourquin

In high school, my English teacher assigned a research paper designed to prepare us for college. For a reason that escapes me now, I chose to write mine on the political and personal satire in *Gulliver's Travels* and *Alice in Wonderland*. I can't really tell you why, but I can tell you that if you think you have issues, you should read up on Jonathan Swift and Lewis Carroll. It took an amazing amount of research, and I do credit the work I put into the paper as good preparation for college. But not just because of the research.

Around 8:00 the night before this fifteen-page beast was due, I headed into the basement where we kept our Tandy TRS-80 (for real) to start writing. I've always worked well under pressure (ironically writing this very StickyLetter the day before deadline), and at that point in my life I enjoyed the occasional all-nighter. So write I did, all night—but save I did not. Around 2 a.m., I had a good structure to the paper and was about twelve pages in, so I decided to print the report and read it on paper. Yet, I did not save the file. Somewhere around page six, my noisy dot matrix printer started chewing on the paper. In my efforts to straighten it out, I pulled the printer plug out of the socket, but it wasn't the printer plug—it was the Tandy TRS-80 plug. With one strike I lost twelve pages. It was an awful moment. Right there I learned the hard way to save and to back up files lest I be stuck with a computer disaster.

Fast forward a few years (OK, a little more) to present day. I have a great personal computer—a MacBook—that I adore. It is full of family photos, videos, music files, the files of my family Web site, and all the other stuff that you put on a personal computer. Because of my love for this computer, I bought it an external hard drive to do regular backups. And I did—once—sometime in 2007. So, two weeks ago when all my MacBook would do is display the international No symbol, I got a little nervous.

I whisked it off to the Genius Bar certain that anything called the Genius Bar would be a place that could solve anything. They tried, even hooking up the hard drive to what looked like a heart-lung machine for computer hardware. As my genius so eloquently put it, my hard drive was "hosed." My MacBook was then shipped off to Texas to be rebuilt, and I walked out of the Apple store empty handed. I was bereft, but mostly I was ashamed. I know better. I had learned the importance of backups and saving early on. Plus, by virtue of the industry I work in, I know better than the average folk. So why, why, why didn't I have a more recent backup? The best answer I can come up with is because, well, I am a nincompoop. After many hours of sifting through CDs, emails, and Web sites, I've reclaimed about 80 percent of the data I lost, but the rest is gone for good and I have no one to blame but myself.

It's never fun to lose something, nor is it fun to admit when you're the root of the problem. But both are valuable lessons. Can you really fix something you've done wrong if you don't acknowledge your part in it? Once I managed to stop feeling sorry for myself for not backing up the hard drive, I spent some time educating myself about the best way to backup valuable data. By the time my rebuilt computer arrived at home, I was ready for it. I now have automatic backups scheduled! The peace of mind is amazing.

What lessons have you learned the hard way? Email me and let me know.

Holly Bourquin  
[hbouquin@sqe.com](mailto:hbouquin@sqe.com)

At the risk  
of heresy, some  
executives are probably no  
smarter or experienced  
than you. Scary, isn't it?

PAGE  
35



## Books Guide

The STICKYMINDS.COM BOOKS GUIDE is one of the most popular areas on our Web site. With more than 880 books—including many that have been reviewed by thought leaders, industry experts, and your peers—the STICKYMINDS.COM BOOKS GUIDE is your first stop for finding a good read. Not sure what you're looking for? Browse books by topic, including:

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

## NOVEMBER WEEKLY COLUMNS

Every Monday we bring you fresh ideas to jump-start your work week. This month we deliver thought-provoking articles from talented authors such as Dion Johnson, Esther Derby, Clarke Ching, and Jeff Patton.

## BOOK REVIEW

**Advanced XML Applications from the Experts at the XML Guild**

**By:** The XML Guild

**Reviewed By:** Dmitri Ilkaev

This book is the first collaborative effort of the XML Guild, a consortium of some of the best independent XML consultants in the world. All members have extensive experience in XML and markup technologies and are actively involved in the establishment of standards and best practices. Numerous members have authored books, articles, and papers.

This book provides a set of learning resources most helpful to those who are already proficient in XML and need expert-level advice. It is not intended to be another exhaustive XML bible but a collection of advanced tips and techniques that the authors have used in the real world. Each chapter is written by the guild member considered to be an expert on a particular topic. Every section comes with the comprehensive set of code samples, diagrams, and references to Web sites where a reader can find additional information on the subject. Increased difficulty of the examples, analysis of pros and cons of the proposed solutions, as well as an overview of the applicability of the number of described approaches are among many advantages of the book. Code samples include not only Java but also .NET and C#.

The book brings the power of XML to light to help readers think about XML in new ways and with advanced applications of old and new technologies.

Visit [www.StickyMinds.com/bythebook10-9](http://www.StickyMinds.com/bythebook10-9) to post your comments on this book.



## WEB SEMINARS

by StickyMinds.com & Better Software magazine

**StickyMinds.com & Better Software magazine Web Seminars:  
One Hour of Brain-Boosting Power**

Performance testing, virtualization, quality assurance, and test-driven development are some of the topics experts in the software industry have covered in past StickyMinds.com and Better Software magazine Web seminars. If you missed a live presentation, check out the Web seminar archive on SQE.com today!

[www.sqe.com/Media/WebSeminars/Archive.aspx](http://www.sqe.com/Media/WebSeminars/Archive.aspx)



# The Award-Winning-Est Agile Lifecycle Management Solution



Three-time Jolt Product Excellence award winner in 2006, 2007 and 2008 for project management tools



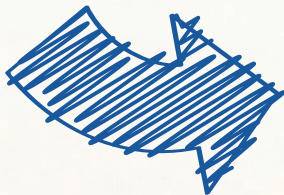
QSM concludes—

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



Forrester Research says—

"Rally designed the requirements management capabilities in Rally Enterprise, a software-as-a-service (SaaS) ALM solution, to suit teams using Agile processes. The product performs flawlessly in this regard..."



Get your FREE trial of Rally Enterprise at

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

No download, no installation, no commitment



Visit us at the Agile Development Practices Conference, booth #200  
November 10 - 14, Orlando, FL



## Testing Services

Are your applications **scalable** enough to meet your **business needs ?**

“

In many organizations, performance testing of applications is done in an ad-hoc manner with minimal test planning. In some cases, performance testing requirements are vaguely defined leading to inconsistency and poor performance at peak loads.

”

## The *right approach* to performance testing

Effective performance testing of applications mandates clear understanding of performance testing requirements. Capturing the right performance statistics both at client side and server side is critical in identifying and understanding the bottlenecks.

As a first step, you will need to identify and prioritize business critical applications for performance testing. A structured performance assessment will help you evaluate the current state of performance testing in your organization. You can thus envision a desired state by benchmarking industry standards and draft a transition roadmap that would help you consolidate and establish an efficient performance testing group. The group can refine scripting standards and guidelines, design work-load scenarios and select the right set of monitoring tools and metrics.

To ensure that your applications are truly responsive you could explore options such as early performance testing and architectural benchmarking.

### Cognizant makes it simple

Cognizant with an expert group of more than 600 performance testing professionals has provided to clients, a wide range of high-value services comprehensively testing for the scalability of applications using industry-standard and open source performance testing tools.

**Performance Assessment** - Cognizant's comprehensive performance assessment framework helps clients identify the gap between the current state of performance testing and the ideal state, thereby aiding construction of a roadmap towards achieving desired performance levels.

**Nitro Booster** - Cognizant's customized performance testing framework for HP LoadRunner provides additional utilities like script modeling, custom function library and custom report generation ability.

**JVigil** - is an agent-less monitoring tool built by Cognizant that is compatible with all JMX compliant, J2EE based application servers.

*Cognizant Testing Services is an "Independent Verification & Validation" (IV &V) service from Cognizant Technology Solutions Inc. With a team of more than 8800 dedicated software testing professionals, Cognizant is the world's largest dedicated testing service provider. Cognizant provides end-to-end testing services to over 250 clients across various geographies and industry domains.*

To know more about Cognizant - the world's largest testing service provider, please log on to [www.cognizant.com/testing](http://www.cognizant.com/testing)



**Cognizant** | Testing Services  
Passion for building stronger businesses



# Follow the Process

by Lee Copeland

Politicians often wrap themselves in their countries' flags, instantly attacking the patriotism of anyone who disagrees with them rather than dealing with difficult issues in a substantive way.

We have our own "flags" in the software development world. Years ago, when I worked in a large mainframe data center, the collective wisdom was "You'll never get fired for buying IBM." No matter what disasters occurred, no matter how inept your technical skills, no matter how poor your management decisions, you were safe—because you bought from the right vendor.

Later, when I led a software development group, we had another "flag"—our software development methodology. While my copy has turned to dust after thirty years, you can find an equivalent methodology, the US Housing and Urban Development's (HUD) 354-page tome, at [www.hud.gov/offices/cio/sdm/devlife/sdm6\\_05.pdf](http://www.hud.gov/offices/cio/sdm/devlife/sdm6_05.pdf). As is common in these methodologies, it is full of exhortations but contains little actual "how to" guidance. For example, in the chapter titled Design System, the HUD methodology commands, "the functional and data requirements ... are further refined to low-level specifications," "specifications are then organized in a way suitable for implementation," and "the project development team uses the functional and data requirements ... for partitioning the system into subsystems." All good ideas, but there is no guidance whatsoever on how to actually accomplish these complex tasks. (I've always been amazed that people write this stuff down in such detail. Which part are they going to forget if it's not in the methodology—requirements, design, coding, installation?)

With thousands of others, we stood together and sang the process song:

"Follow the process, follow the process, follow the process, don't go astray.

Follow the process, follow the process, follow the process, it knows the way."

And, no matter what disasters occurred, no matter how inept your technical skills, no matter how poor your management decisions, you were safe—because you followed the process.

Then came the mother of all software development methodologies—the Capability Maturity Model at 450 pages, only to be followed by the über-mother, the Capability Maturity Model Integrated at 700 pages.

And, with thousands of others, we stood together and sang the process song:

"Follow the process, follow the process, follow the process, don't go astray.

Follow the process, follow the process, follow the process, it knows the way."

And again, no matter what disasters occurred, no matter how inept your technical skills, no matter how poor your management decisions, you were safe—because you followed the process.

Now, a new set of methodologies has emerged under the umbrella called "agile." Born of a rejection of the heavyweight methodologies, we have Extreme Programming, Scrum, Crystal Clear, DSDM, adaptive software development, feature-driven development, behavior-driven development, the Agile Unified Process, and a host of others.

While very different from the heavyweight methodologies, these agile practices are also methodologies—"a set of practices, procedures, and rules used by those who work in a discipline or engage in an activity." And, as such, these methodologies give little "how to" guidance. For example, in *Extreme Programming Explained*, Kent Beck states, regarding testers, "You are responsible for helping the customer choose and write functional tests"—a great idea, but that statement

A methodology  
is a roadmap,  
and roadmaps can be  
misleading.

doesn't tell us how to actually choose and write these tests.

And, with thousands of others, today we stand together and sing the process song: "Follow the process, follow the process, follow the process, don't go astray.

Follow the process, follow the process, follow the process, it knows the way."

And again, no matter what disasters occur, no matter how inept your technical skills, no matter how poor your management decisions, you will be safe—because you followed the process.

I have no quarrel with any of these methodologies. I'm sure in an appropriate context, each is excellent for guiding the creation of better software. It's all the dang singing that I can't stand.

A methodology is a roadmap, and roadmaps can be misleading. First, the methodology may be flawed in that it is simply wrong about some things and omits others. Second, none of these methodologies tells us *how* to do the tasks—how to elicit requirements, design, code, and test (the really difficult and interesting stuff). Third, all of these methodologies teach us to resist change. The methodology is the only true path—there is no other.

"Follow the process" leaves no room for "tap into your wisdom" or "listen to your intuition" or "use your experience." It also omits the idea of "take personal responsibility" for success, substituting blind process obedience. Your head, heart, and hands are the keys to your success in building better software—not the methodology. **{end}**

# Encapsulation and Vampires

by Kevlin Henney

Encapsulation—it’s all about declaring data in classes private, right? Well, not quite.

Encapsulation has a deeper meaning with more profound implications for your code than just “declare your data private,” although this is what text books and training courses have dumbed it down to mean. Consequently, in considering encapsulation, many programmers look no further than the presence or absence of the keyword “private.” Encapsulation is a matter of degree and sensibility; it is not simply a binary state enabled by sprinkling `private` over your code.

What, then, does it mean to encapsulate? Perhaps the most useful and revealing definition is to be found not in a book on software development but in an ordinary dictionary, the *New Oxford Dictionary of English*:

## encapsulate

- Enclose (something) in or as if in a capsule.
- Express the essential feature of (someone or something) succinctly.
- Enclose (a message or signal) in a set of codes which allow use by or transfer through different computer systems or networks.
- Provide an interface for (a piece of software or hardware) to allow or simplify access for the user.

So encapsulation involves a boundary, the succinct expression of something’s essence as an abstraction, and interposition of an interface to simplify access. Data privacy is simply one effect of encapsulation, but it is not the only one. And, as an effect, it should not be confused with a cause.

## Private Parts

Let’s consider an example problem from a previous Code Craft column (“Programming with GUTs” July/August 2008), that of a recently used list. However, rather than use the example as a means of focusing on unit-test quality, let’s take a slightly different journey. For variety, let’s walk through it in C++ rather than C#. Using C++ gives us the excuse to start off in C+ (C++ used as a better C) and allows us to take in a couple of other sights en route. The conclusions drawn apply to the same coding habits when practiced in other languages, but C++ serves to amplify and emphasize the choices made.

To start with, we can consider a recently used list to be a sequence of strings. Everyone’s favorite default sequence container is `vector`, which gives us the modest starting point in listing 1.

The pro: We’ve given the concept a name by which it can be used. The con: It doesn’t actually behave like a recently used



ISTOCKPHOTO

```
typedef std::vector<std::string> recently_used_list;
```

Listing 1

```
struct recently_used_list
{
    void insert(const std::string & most_recent)
    {
        std::vector<std::string>::iterator found =
            std::find(
                elements.begin(), elements.end(),
                most_recent);
        if(found != elements.end())
            elements.erase(found);
        elements.insert(
            elements.begin(), most_recent);
    }
    std::vector<std::string> elements;
};
```

Listing 2

list. Recently used lists aren’t just arbitrary sequences; they are both stack-like and set-like. They maintain an order of insertion, but they also preserve the uniqueness of their elements.

Listing 2 shows a refinement that attempts to take this into account. The pro: We have associated an operation with the

```

class recently_used_list
{
public:
    void insert(const std::string & most_recent)
    {
        std::vector<std::string>::iterator found =
            std::find(
                elements.begin(), elements.end(),
                most_recent);
        if(found != elements.end())
            elements.erase(found);
        elements.insert(
            elements.begin(), most_recent);
    }
    std::vector<std::string> list() const
    {
        return elements;
    }
private:
    std::vector<std::string> elements;
};

```

Listing 3

data type, an operation that preserves uniqueness on insertion and would otherwise be repetitive, tedious, and error prone to write against the typedef in listing 1. The cons: We have failed to capture any of the other constraints or behavior that makes a recently used list a recently used list—as opposed to just a resizable array. We have allowed operations, such as inserting arbitrary values in arbitrary places, that make sense for vector but not for a recently used list.

Furthermore, even though we have a member function, we are still using C's `struct` rather than C++'s `class`, and we have yet to use the `private` keyword. That must be the problem! Listing 3 addresses this “problem” in a way that turns out to be quite common in practice.

We now have a capsule in the form of the `recently_used_list` class. The separation between declared public and private parts defines a boundary. The class captures some of the essential features of a recently used list, but whether it does so succinctly, completely, and efficiently is another matter. We have simplified the act of insertion but not much else. This qualified assessment suggests only qualified success. Although many programmers would consider this code to be sufficiently encapsulated—and, therefore, their work complete—this style of class design is almost as unencapsulated as the code in listing 2. And we've only just started.

## Citing References

In C++, by default, function arguments and results are passed by copy. For simple types, such as integers and pointers, this is both unsurprising and cheap. However, in returning elements every time the `list` function is called, we are copying an array that holds strings, each of which is also copied. Given the limited capabilities of `recently_used_list`'s public interface, the `list` function is likely to be one of the most heavily

```

class recently_used_list
{
    ...
    const std::vector<std::string> & list() const
    {
        return elements;
    }
    ...
};

```

Listing 4

called functions. This could prove expensive.

So, in listing 4 we have returned the result by reference, bypassing the copy, and have retained source compatibility with all the dependent code that would have been meaningful against listing 3. The `list` function retains its query status, marked with a trailing `const`, but now returns a `const`-qualified reference to elements. This ensures that the caller can look at the elements but can't change them.

On the one hand, it is important not to become overly obsessed with micro-optimizations. These often distract programmers from seeing the bigger picture. On the other hand, it also makes sense to avoid coding habits that squander resources and introduce gratuitous pessimizations. How gratuitous? In comparing the relative performance of using the code in listing 3 with that of listing 4—for example, to call the `size` function on the result of `list`—you can be looking at a difference of two orders of magnitude in execution time.

In this example, another motivation for returning by reference rather than by copy is to satisfy the principle of least astonishment. The code in listing 3 blows up quite spectacularly should you try to iterate over the contents by calling iterator operations on the result of `list`. Each iterator returned will be traversing against its own temporary (and deceased) copy of the content.

So, the code in listing 4 is not only more efficient, it is safer. But is it better encapsulated? Possibly, given the improved usage experience, but only possibly. A trade has been made that may make you feel a little uncomfortable. We have committed the internal implementation to being a vector. If we were to choose another container, such as a `list` or a `deque`,

```

class recently_used_list
{
    ...
    const std::vector<std::string> & list() const
    {
        return elements;
    }
    std::vector<std::string> & list()
    {
        return elements;
    }
    ...
};

```

Listing 5



the result type would have to be changed. If returning containers by copy is somewhat questionable, hardwiring the returned container type by reference to an arbitrary internal type is even more suspicious.

Let's amplify this discomfort a little further by following a path taken by many programmers, which is certainly the default path when following this style of class design in languages such as C# and Java. Listing 4 provides read-only access to the internal implementation. Good, except that we are prevented from performing convenient operations that modify the recently used list such as calling `clear` on the result of `list` to clear it of all of its entries. We can "solve" this by overloading the `list` member function so that when called on a plain `recently_used_list` (as opposed to a `const-qualified` one), it returns an unqualified reference to a vector. The class in listing 5 now supports the ability to clear a recently used list via the `list` function. It turns out, however, that this is a pyrrhic design victory.

## Beware of Vampires

To appreciate why we have blown out of the water any encapsulation we may have had, let us briefly consider vampires—specifically, vampire lore as popularized by Hollywood in movies such as *The Lost Boys*, *Blade*, and *Underworld* and in TV series such as *Buffy the Vampire Slayer* and *Angel*.

How do you protect yourself from vampires? Garlic, holy water, crucifixes, sunlight (often just UV light will do), or a stake through the heart are among the answers normally given. One simple form of protection is often overlooked. To quote from *The Lost Boys*, "Don't ever invite a vampire into your house, you silly boy. It renders you powerless." Without an invitation, a vampire cannot cross the threshold.

And our class? Returning a reference to the internal state qualifies as an invitation. The code is saying "Bite me." The caller, intentionally or otherwise, can now break the good form of the object, violating invariants such as uniqueness. Dependent code is now coupled to and feeding off internal implementation details. So much for `private` meaning *encapsulated*.

The take-home lessons of this column extend further than just the specifics of the example. They should

help to drive a nail into the coffin of the vampiric but ever-popular setter-getter style, a coding style that is perhaps best characterized as object-oriented assembler. **{end}**



**What code have you come across that invites in the vampires?**

**What problems have been caused and how have you been able to address them?**

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

## WHEN IT'S TIME



## TO OUTSOURCE SOFTWARE DEVELOPMENT



## IT'S TIME TO CALL ACULIS YOUR ONE-STOP TECH SHOP.

On-site, off-site & off-shore — Our customers rely on us to provide an integrated approach to custom software development, comprehensive quality assurance & software testing, high quality globalization & translation solutions, and flexible IT staffing & recruitment. On target, on time, on budget, contact ACULIS to accelerate your IT development and maximize your ROI.

**ACULIS** / IT ACCELERATOR  
DEVELOP IT / TEST IT / GLOBALIZE IT / STAFF IT

1-866-4ACULIS / WWW.ACULIS.COM

# BETTER SOFTWARE

CONFERENCE & EXPO

JUNE 8-11, 2009  
LAS VEGAS, NEVADA  
*The Venetian*

KEYNOTES • TUTORIALS • WORKSHOPS • CLASSES

Agile Development ↑

Project Management ↑

People & Teams ↑

Testing & QA ↑

Requirements ↑

Process & Metrics ↑

Design & Architecture ↑

*Over 98% of 2008  
Attendees Recommend Better  
Software Conference & EXPO  
to Others in the Industry*



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

# Cover or Discover?

by Michael Bolton

Most of the time, excellent testing and our evaluation of the quantity and quality of our test coverage start with asking who our client is and what he wants to know about the system. Yet, if our client isn't knowledgeable about testing, development, or some other aspect of the context in which the product must work, it's possible that neither we nor the client knows what he wants to know. So where could we start? We might want to start by looking at a map of the system and asking questions about it. Suppose that I showed you a map of a subway system and asked you to cover the map. How would you do it?

You might start by choosing a station at the end of one of the lines on the system and then riding the train to the other end of the same line. Then you might do this for each of the other lines in turn. When you had covered all of the lines, would you have achieved complete coverage of the map?

Maybe not. Most subway lines consist of two or more tracks in each direction. To cover the system, you'd have to take each line in both directions. So you do that. Would you be done then? Maybe not. There are also sidings off the main lines, so that trains can turn around or pass one another. After you have visited all the sidings, would you have achieved complete coverage? Subway lines don't exist on their own; there are transition points between the lines. To cover the map, you'd likely want to cover the interchanges between the lines in every direction. And is it sufficient merely to travel the lines, or would we need to stop at each station and have a look around?

So far we've been thinking about covering the map in an abstract, structural way, but we haven't been thinking very much about what actually happens on or with the system. We haven't talked much about the rolling stock—the trains themselves or other vehicles like maintenance wagons. We have yet to discuss the infrastructure—track, signaling systems, elec-

trical supply—that is needed to support the operations of the system. We could look at the interconnections with other systems—sidewalks, buses, streetcars, and commuter train services. We might talk about the different types of people who use the system—commuters, students, and people with strollers or in wheelchairs. We might consider why they are using the system—to get to work or school during peak hours, to get to cinemas and art galleries on the weekends, or to get home after the bars close. We could discuss aspects of the system that might seem peripheral or incidental to its primary purposes—signage, collection booths, janitorial services, newsstands, and advertising. We should consider the relationship between the system and time—scheduling, actual time between trains, boarding time. We haven't considered how our observations might be affected by the times at which we make them—at rush hour, in the middle of the day, in the middle of the night when the system is closed. And then, we could think about combinations of circumstances and how they might result in behavior that is non-linear—at rush hour, trains are more crowded, so they take longer to board, so the trains can't leave the stations as quickly as at other times, so the system backs up, so the trains become more crowded.

Maybe we haven't talked about these things because we—or our clients—haven't completely realized the motivation for why we're testing. Yet maps rapidly inspire thinking about what we might think or do to cover them and raise possibilities about what we might see.

Some representations of a system actually look like maps—flowcharts, diagrams, UML models, and the like. Usually we design tests by asking ques-



ISTOCKPHOTO

tions about specific things that are labeled on the map such as the steps in some task that the map models. But if we're stuck for ideas on what might be most important to cover, we could start with some map and ask questions that apply generally to maps or apply some general guideword heuristics—*simplest, popular, critical, complex, pathological, challenging, error handling, periodic*. “What's the easiest way to get from here to there? What's the *standard* way? What routes are essential? What parts of the route are potentially confusing or error prone? What could interfere with this task? What would happen if this route were missing or compromised? Does the map suggest what workarounds might be available? How much traffic goes along this path? How fast does it go? Are there possibilities for collisions?”

But another class of questions might arise, too, based on the idea that we might add details to an existing map. An older version of the Toronto subway map that I have indicates the lines and their connections, the stations, and interconnections with suburban rail. A newer version adds extra labels—the street addresses of the subway stations and the locations of parking lots, public restrooms, elevators for people in wheelchairs or with strollers, and surface route transfer points—but drops the labels for the suburban rail transfer points. This reminds us that any map includes some kinds of information that might be useful



and leaves out other information. Comparing and contrasting two maps may suggest ideas that neither map covers.

Some suggest that a map must accurately reflect the territory in order to be useful, but there are plenty of reasons to think otherwise. Subway maps often display the relationships between stations but not the actual distances between them. A story in *Sensemaking in Organizations*, by Karl Weick, suggests a more profound example of usefulness despite inaccuracy. A Hungarian Army unit is on patrol in the Swiss Alps. A big storm comes up, and one platoon doesn't return to camp for a day, two days, three days. The lieutenant is now panicked ... and suddenly the entire platoon walks back into camp. "We thought you were lost for good! Where have you been?" "Well, when the storm came up, we took shelter. When the weather cleared, we realized that we were lost. One of us had a map, though, so we looked at it, and we realized that if we went down the hill, we'd hit a river, and if we followed the river, we'd get to the town ... and here we are!" The lieutenant looked at

the map and realized to his surprise that it wasn't a map of the Alps, but of the Pyrenees. Says Weick:

This raises the intriguing possibility that when you're lost, any old map will do ... maybe when you are confused any old strategic plan will do. Strategic plans are a lot like maps. They animate and orient people. Once people begin to act, they generate tangible outcomes in some context, and this helps them discover what is occurring, what needs to be explained, and what should be done next. Managers keep forgetting that it is what they do, not what they plan, that explains their success. They keep giving credit to the wrong thing—namely, the plan—and having made this error, they then spend more time planning and less time acting. They are astonished when more planning improves nothing [1].

It's easy to see how covering a map might be useful. Ultimately, though, it's

what people think and what people do that make the difference. Analyzing a map and planning how to cover it—test design—can suggest ideas that the map on its own doesn't cover. That might spark us to annotate the map we've got or to create a new one. Comparing the map to the territory—test execution—takes us places where we learn things, even when the map is limited or inaccurate. Excellent testing isn't just about covering the map—it's also about exploring the territory, which is the process by which we discover things that the map doesn't cover. {end}

#### REFERENCES:

[1] Weick, Karl E. *Sensemaking in Organizations*. Sage Publications Inc., 1995. p 54-55.



**Of course there are ways to learn about the product other than maps or diagrams. What representations do you use and how do you cover them efficiently?**

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

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

## DON'T BE SHY

about transitioning to agile.

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

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

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

### Upcoming Courses

#### Dallas

**January 27-28**  
Certified ScrumMaster

**January 29**  
Agile Estimating and Planning

#### Boulder

**February 18-19**  
Certified Scrum Product Owner\*  
\*with Ken Schwaber

#### Seattle

**March 31-April 1**  
Certified ScrumMaster  
**April 2**  
Agile Estimating and Planning

#### Orlando

**May 12**  
Effective User Stories for Agile Requirements

**May 13-14**  
Certified ScrumMaster

**May 15**  
Agile Estimating and Planning



**MOUNTAIN GOAT**  
SOFTWARE

# Keep Non-developers in the Loop

by Melanie Tayler

“What the heck is this?” Corey, senior developer on the WCH project, said as he charged into Lynn’s office. His face was as red as his Ottawa Senators home jersey. “I can’t believe this,” he said, slamming down a sheaf of bug reports from our tracking system that he had wasted trees printing out. “All of these bugs were reported by Alison.”

“Let’s have some coffee and talk about it,” Lynn said, watching the pulsing vein on Corey’s forehead. “On second thought, maybe you’ve had enough coffee.”

Corey sat, but his knee still bobbed rhythmically. “I don’t have time for this.”

“What’s the problem?” Lynn asked, looking at the bug reports.

“QA is the problem. None of these are bugs—and now guess who gets to deal with them?”

“If they aren’t bugs, why did Alison report them?”

Corey’s knee started to bounce double time. “I don’t know. Her probation isn’t up yet, is it?”

Lynn ignored that comment. “There’s a problem here, and we’re going to solve it.”

Not trusting Corey to play nice in his caffeinated state, Lynn IMed Alison, a new hire in the QA department. “Alison,” she typed, “it seems that the bugs you reported on Corey’s recent patch aren’t bugs.”

“I don’t understand,” Alison replied. “I based the test cases on the spec I was given last month.”

“The spec?” Corey rolled his eyes when Lynn told him Alison’s response. “We’re agile.”

“It seems our communication plan isn’t so agile. The spec changed and QA doesn’t know?” Lynn asked Corey.

“Our milestone is next week. We don’t have time to update the spec. The new functionality was decided from a problem exposed by Bob during last Monday’s scrum meeting.”



ISTOCKPHOTO

“And QA doesn’t go to the scrum.” Lynn had seen the five-minute daily update meetings in the lounge, and there was no one from QA in sight. “So how are they supposed to write an up-to-date test plan? How can we encourage developers to update the specs?”

Corey shrugged. “I guess we could put it on the wiki instead of in a Word doc. That would make it easier for the spec to be more of a living document.”

“That’s great. That way anyone can see a delta of the changes at any time, plus the QA team members can ‘watch’ the page so that they get an email when it changes. Is QA on the project mailing list?”

Corey nodded. Originally, the lists had been set up for code review purposes, but they’d evolved so that a lot of the team communication went through them.

“Why don’t the QA people just update the spec themselves?” Corey huffed.

Lynn paused. Why don’t they? It would ensure that QA was in the loop, and the specs might be kept more up to date. Would the QA team be willing to take on that responsibility?

“That’s a good idea. I’ll talk to Olivia about it.” Lynn jotted a quick reminder to schedule a meeting with the

QA project leader. “One thing, though, if they don’t know the functionality’s changed, how can they update the spec? Why don’t QA team members come to the scrums?”

Corey wriggled in his chair like he couldn’t quite get comfortable. “It’s too technical. They wouldn’t get anything out of it.”

“Hey, we make an effort to hire pretty technical QA people around here.” It was a lesson they’d learned some time ago. “I think they might be able to identify a functionality change when they hear one. Besides, it could help the QA team develop more technical expertise over time and maybe even to write better test plans. Let’s try it.”

A month later, Lynn met with Olivia and Corey to see how the new system was working.

“Not bad,” Olivia said, sipping her green tea. “But it could be better.”

Corey seemed a lot calmer than the last time Lynn had met with him. “There are still some problems. They miss stuff.”

“That’s one way of putting it,” Olivia said. “Another way is that it’s hard for us to capture changes that we don’t know about. We aren’t always kept in the loop.”

“What do you mean?” Lynn asked.

## STORY LINES

- **Functionality changes that aren't reported to QA can result in time-wasting bugs.**
- **Keep the QA group in the loop by letting it help keep the spec up to date.**
- **Inviting QA members to development scrums and meetings keeps them up to date on changes and helps them gain technical knowledge.**
- **Consider putting the specs on an internal wiki to help them become a living document.**

"Well, a lot of changes are decided on the fly."

"If clients want to change something, they need to be able to," Corey said.

"Of course," Olivia agreed. "But just tell us about it."

Corey shrugged. "The code is the priority."

"In other words," Lynn thought, "they don't think about telling QA."

"I hate adding more process," Lynn said. "But we might need to."

Olivia broke in. "What if you cc me when you're telling the client or other developers about the change? I can take care of forwarding the info to the right person on my team."

"That would really benefit development," Lynn pointed out. "QA's test cases could then concentrate on finding real bugs."

Lynn had her doubts that development would make that change overnight. She'd seen that QA team members were now attending scrums, but she had the feeling that the new attitude would take time.

"And, there's another problem," Olivia said, shaking her head. "I have to be honest with you. We can't always document some of the more technical

things, such as network protocols or API changes, so we can't be solely responsible for updating the specs. But if we added a section in the project leader's weekly status report to identify those changes, it would warn QA not to write test cases based on outdated info."

"But the status only goes to the client," Corey said. His eyebrows knit together.

"Send it to the mailing list," Olivia suggested.

"I guess that'd be OK. If we want to go big, we could use the wiki to track the spec changes so you don't have to search the mailing list every time you want to find one. As they come from the stakeholder, we could even track the requests in a table that shows the request, who requested it, and the request status."

"When does QA need to know that a code patch implements a spec change?" Lynn asked Olivia.

Olivia took another sip of tea. "Before we have to test it. Ideally, with enough time to identify and adapt all affected test cases or write new ones. We

just want to be confident that the test plans reflect what needs to be tested."

"We already have sections in our code patch submissions that summarize changes and unit test results. QA already reviews this, so what if we also include a section that tracks when a patch implements a requirement change?" Lynn suggested.

"I guess that wouldn't add too much to the process," Corey said.

"You know," Lynn reflected, "he's really been a lot easier to deal with since he switched to decaf." **{end}**



**What methods have you used to keep non-developers in the loop?**

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



## The Award-Winning-Est Agile Lifecycle Management Solution



Three-time Jolt Product Excellence award winner in 2006, 2007 and 2008 for project management tools



QSMA concludes—

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



Forrester Research says—

"Rally designed the requirements management capabilities in Rally Enterprise, a software-as-a-service (SaaS) ALM solution, to suit teams using Agile processes. The product performs flawlessly in this regard..."



Get your FREE trial of Rally Enterprise at [www.rallydev.com/bsm](http://www.rallydev.com/bsm)

No download, no installation, no commitment



Visit us at the Agile Development Practices Conference, booth #200



A glass of water sits on a parched, cracked earth surface. The background is a vast, dry landscape under a bright blue sky with scattered white clouds. The glass is partially filled with water, and the cracked earth is visible through the water.

# *GETTING* **AGILE** with USER-CENTERED DESIGN

**CREATE SOFTWARE  
USERS CAN'T LIVE WITHOUT**

by  
JON DICKINSON &  
DARIUS KUMANA





**T**he agile software development movement has made huge improvements in reliability when delivering software, increasing return on investment, and reducing the risk of building software. However, in a world of iPhones and Google apps, this may no longer be enough.

People are beginning to expect more from software. They expect it to work—not work in the sense that the software won't crash every half an hour, but work intuitively and hassle free, as if it were built just for them. As users become more computer savvy, they have a better understanding of how they expect programs to behave. More often than not, users know what they want to achieve. Any software that hinders them from efficiently achieving their goals will quickly be replaced. This is particularly true where users can exercise their freedom of choice between a number of alternatives, such as with Internet-based applications where a viable alternative is only a Web-search away. If you are building this kind of software, you need to tune in to your users quickly.

There is no doubt that to continue to provide value for our customers, we must continue to apply the principles of the agile development philosophy. But, in order for our software to be truly successful in the eyes of its biggest critics, we must endeavor to adopt a more user-centered approach.

### **What Is User-Centered Design?**

User-centered design (UCD) can be applied to the design of anything that has a user—from mobile phones to kitchens. But when integrating UCD with agile practices, we are only concerned with the arena of software development. With agile development, the primary measure of progress must be related to *working software*. Once you adopt a user-centered philosophy, this is no longer the whole story.

Unlike agile, UCD is not focused on the customer—it is centered on the end-user. Furthermore, from a UCD standpoint, software is incidental; what is important is that end-users can easily and efficiently achieve their goals—with or without software.

### **Why Do We Need User-Centered Design?**

There are a number of benefits that arise when advocating a user-centered phi-

losophy. User-based research provides a mechanism against which design decisions can be validated and tested. Evidence-based decisions mean that guess work is minimized. What to build becomes much less of a matter for debate. More importantly, by keeping a product's end-users at the heart of its design and development process, the end result is far more likely to be useful, usable, and meaningful.

The era of feature-centric development is coming to an end. Consumers are beginning to realize that more features do not always mean a better product. In the current maturing marketplace, quality of experience is far more likely to be a product differentiator than number of features—think iPhone vs. Nokia or Wii vs. PS3.

UCD provides a way to engineer these quality experiences. As such, it empowers development teams to create products and solutions that are competitive in today's discerning market. By embracing a UCD philosophy, one could argue (as Peter Merholz does in "Experience Is the Product" [1]) that we should not just create products and services—we should aspire to build better overall experiences where the value (both quantitative and qualitative) to all concerned is obvious.

### **What Is Agile Software Development?**

Agile software development is a philosophy that provides a framework defined by a set of values and principles that attempt to focus software developers on delivering the most value possible to their customers.

### **Why Do We Need Agile Software Development?**

Building software is a tricky business. There are a number of factors that contribute to this complexity:

- Writing software is technically complex and mentally challenging.
- People write software, and people do not perform in a consistent manner.

- When software is commissioned, people might know the broad problem to be solved but do not understand the intricate detail of how software can help.
- Software takes a long time to implement, and the context of the problem domain can change while the software is being built.

Agile recognizes these issues and provides some guidelines to help mitigate the risk they cause. When working to the agile principles, we focus on delivering software to the customers as early and as frequently as possible, allowing them to give feedback, validate, and adjust their perception of what it is they *actually need* from the software.

## Commonalities Between UCD and Agile

A striking but sometimes overlooked similarity between agile and UCD is that both are often fundamentally misunderstood by people starting out on the path of adoption. This misunderstanding stems from the fact that both are frequently assumed to be *methodologies*—magical, step-by-step recipes that you can follow to guarantee project success. In fact, agile and UCD are both *philosophies*.

There are many different methodologies that implement the agile philosophy: Extreme Programming, Scrum, the Crystal family, etc. There are also several different interpretations of UCD. User-experience aficionados can learn from the way products and experiences are created by the likes of Adaptive Path, Cooper, Apple, Shedroff, Morville, Spool, Nielsen, and Norman (see the StickyNotes). While the specific methods are very different, the underlying philosophies are undeniably similar.

Both the agile and the UCD philosophies are iterative; they progress in small steps providing opportunities for verification and refinement along the way.

## The Conflict Between UCD and Agile

There is a history of conflict between agile developers and user-experience designers. Agile software development is predominantly a developer-led phi-

losophy, while UCD is, in many organizations, championed by creatives. There will always be a healthy, natural tension between these left-brained and right-brained individuals.

If we examine the principles behind the agile and UCD philosophies, then we are much more likely to find tangible issues that can be addressed to facilitate agile developers and UCD practitioners working together to create quality software and robust user experiences, rather than falling back on the age-old argument that developers are from Mars; designers are from Venus.

To better understand the disconnect, let's compare some of the principles of the Agile Manifesto that cause conflict with the UCD philosophy.

*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*—Agile Manifesto

The Agile Manifesto focuses directly on providing value for the customer, whereas UCD champions the end-user—the idea being to create software that users “cannot live without.” By delivering such intuitive software we cannot fail to deliver knock-on value to our stakeholders.

The equivalent primary principle for UCD might read:

*Our highest priority is to help create an experience for end-users where they can achieve their goals easily and efficiently with minimal disruption to their mental model of the problem space.*

\*\*\*

*Working software is the primary measure of progress.*—Agile Manifesto

The concern here is the definition of working. The aim of this principle when it was defined was to get software development teams out of the mindset that creating a lot of design documentation before writing any code was helping to meet the project deadline. The goal of a software development project is to produce functioning software, not UML diagrams. From a UCD perspective, software that simply works is a secondary measure of progress. Much more important is whether the software helps the users achieve their goals.

The equivalent UCD principle might read:

*The satisfaction of end-user needs (user goals) balanced with the achievement of business goals is the primary measure of success.*

\*\*\*

*Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*—Agile Manifesto

Agile development calls for early delivery of the software to the customer. This is *not* synonymous with releasing to end-users. The customer's public release strategy can be entirely separate from this process. Early delivery to customers allows beta tests or usability trials to be performed and the customer to realign his priorities based on the findings.

If we release too early to the market, the end-user experience could be poor. In some situations, this could be seriously detrimental to the impact of the product on the market and ultimately the bottom line. For example, in 2005, the early release of handsets that were touted as “feature complete” and “bug free” cost the mobile phone industry \$4.5 billion:

“One in seven mobile phones are returned within the first year of purchase ... 63% of the devices being returned are done so without fault.”

“Most of these issues may be addressed through stringent device testing and usability modeling prior to the launch of the mobile device.” [2]

## Common Issues When Integrating UCD and Agile

### Design without constraints

When the design of a system is created with users and customers but without regular feedback from a development team, there is significant risk of a design being proposed and approved when no one has any idea of how long it will take or how much it will cost to implement. When the estimate does come in, and it's far above the amount expected, it is understandable that the customer will feel let down by the development team. To compound the problem, the design team will often be frustrated that the only way to deliver the software within the real budget will be to “water down” its design. As stated in the Agile Manifesto: *Business people and developers*



*must work together daily throughout the project.*

In this context, it is often considered that the UCD practitioners would operate as part of the customer team [3] to help drive the direction of the product.

## Validate with real-world usage

UCD is an iterative process that calls for designers to validate and refine their design regularly. While the design is being created, many techniques are used to perform this validation that provide value through quick and short feedback loops. Despite this, we maintain that the design of a successful user experience cannot be completed without feedback from people using the real system. This is where we get benefit from the primary agile principle mentioned earlier: *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*

To be able to perform and respond to this level of validation, it is necessary to have the design team integrated with the development team to run the tests and modify its design based on the findings.

## Handover is the enemy of understanding

Whenever one team is responsible for creating something it will hand over to another team, there is a risk that the receiving team won't understand the theory or mental model behind the item. This is a common problem even when the two teams have similar backgrounds, but it is compounded when the teams have very different skill sets and backgrounds, as is the case with development and design teams. When designers want developers to implement their ideas, it is not enough to pass on screen shots or scribbles. There must be a shared understanding of the problem domain and the theory describing the reasons why certain decisions have been made. Like the Agile Manifesto says: *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*

## No time to iterate

A common flaw, articulated by both Alistair Cockburn [4] and Jeff Patton [5], is that on an agile development team

people forget to iterate, or at least forget to plan to iterate.

It is quite common for an agile software development team to forget that the point of all the feedback is to allow customers to inspect and verify the features that they have requested. The result is that it comes as a shock to the customer that valuable features must be postponed when a recently implemented feature needs to be iterated over again.

When taking a user-centered approach to agile development, it is critical that we make time to iterate so we can respond to user feedback without pushing out features that we have told the customers they will receive. There are a couple of strategies to manage this: Take a percentage of your initial estimate and add it to the schedule, or consider Alistair Cockburn's approach of creating three cards for user rights [6] by planning the implementation and two additional iterations of each story.

However you solve this problem, the first step is recognizing that while providing shorter feedback loops to the customer and users is valuable, it doesn't count for much if we don't plan to get the benefit from the feedback we receive. Keep in mind the importance of the agile principle that says we should: *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*

## Divided responsibilities; divided teams

Agile developers often focus more on the delivery of software than on *valuable* software. Following the "you aren't gonna need it" [7] mindset, agile developers run the risk of just developing the minimum of what they are asked without actively taking the responsibility to volunteer information regarding alternative approaches to the product owner or UCD practitioner. We must actively challenge the mindset of divided responsibility—"You spec and design it; we'll build what you spec." Everyone should work toward the shared vision of a successful experience.

The flip side of this is the UCD practitioner or product owner's not actively seeking collaboration from the devel-

opment team and, therefore, not being aware of the technical constraints upon the proposed design.

This is addressed by the agile principle: *Business people and developers must work together daily throughout the project.*

## Conclusion

In today's marketplace, experience is king. In the eyes of end-users, what makes software great is their perceived experience. So how do we build great software? As one would expect, there is no magic formula.

Not only does agile provide us with tangible software sooner but it also provides the transparency and constant feedback against which we can validate and steer decisions. User-centered design can help us remain focused on the goals, frustrations, and desires of our end-users.

A successful combination of these two philosophies, while sometimes painful for the practitioners involved, will result in the creation of product experiences that provide the "wow" factor to captivate the discerning users of today's marketplace. {end}

### REFERENCES:

- [1] [www.core77.com/reactor/06.07\\_merholz.asp](http://www.core77.com/reactor/06.07_merholz.asp)
- [2] [www.wdsglobal.com/news/whitepapers/20060717/20060717.asp](http://www.wdsglobal.com/news/whitepapers/20060717/20060717.asp)
- [3] [www.agileproductdesign.com/blog/emerging\\_best\\_agile\\_ux\\_practice.html](http://www.agileproductdesign.com/blog/emerging_best_agile_ux_practice.html)
- [4] [alistair.cockburn.us/index.php/Incremental\\_versus\\_iterative\\_development](http://alistair.cockburn.us/index.php/Incremental_versus_iterative_development)
- [5] [www.stickyminds.com/s.asp?F=S13178\\_COL\\_2](http://www.stickyminds.com/s.asp?F=S13178_COL_2)
- [6] [alistair.cockburn.us/index.php/Three\\_cards\\_for\\_user\\_rights](http://alistair.cockburn.us/index.php/Three_cards_for_user_rights)
- [7] [c2.com/xp/YouArentGonnaNeedIt.html](http://c2.com/xp/YouArentGonnaNeedIt.html)

## Sticky Notes

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

■ Further reading



# Google Web Toolkit

## Writing Ajax Applications

## Test First

by **Daniel Wellman**

**NOTE:** Just before press time, Google released Google Web Toolkit (GWT) version 1.5, which supports Java 5 language features such as generics, annotations, and enumerated types. GWT 1.5 includes numerous other enhancements such as compiler improvements to improve the speed and size of generated JavaScript, overlay types to easily wrap JavaScript objects, animations on the standard widgets, and many more.



**R**achel had successfully built her Web 2.0 Ajax application using Google Web Toolkit, and her boss and customers were ecstatic. Revenue had increased, which led to a steady stream of new business requirements.

But there was a catch: Bugs appeared, got fixed, and mysteriously reappeared. Rachel found her team spending an increasing amount of time fixing bugs while new feature development slowed to a crawl. She remembered that test-driven development was a great way to build applications in a manner that prevented defects from reappearing. She knew that Google Web Toolkit had been built with testability in mind, so Rachel downloaded JUnit and started writing tests ...

In the October 2008 issue of *Better Software* magazine, part one of this series introduced the Google Web Toolkit (GWT), a tool for building cross-browser Ajax applications written in Java. GWT compiles Java code into JavaScript and provides a component library that is cross-browser compatible and memory-leak proof. This means that you can focus on writing your application business logic instead of handling the accidental complexity of supporting multiple browsers.

Another core feature of GWT is *testability*, which means it's easy to unit test your application. This makes it possible to write GWT applications *test first*—an agile practice that helps build reliable and extensible applications. This article introduces GWT's testing infrastructure and demonstrates how to build an Ajax application test first.

## GWT's Testing Infrastructure

Since a GWT application is almost entirely written in Java, you can test almost all of it using standard JUnit tests. However, GWT also includes a special subclass of JUnit's `TestCase` that can test code that requires JavaScript at run time. While all of your client-side Java code will ultimately be compiled to JavaScript, only some of it directly uses code implemented as JavaScript. For example, the code in listing 1 is from the `GWTHTMLTable` class.

This code sample demonstrates a method written in Java (`setStylePrimaryName`) that relies on code implemented

directly in JavaScript, as indicated by the keyword “native” in the definition of `getCellElement`. As shown in listing 1, many of the GWT libraries include some native code—in particular, all widgets that manipulate the Domain Object Model (DOM). Thus, when your unit tests execute native JavaScript, you must be running in an environment where it can be executed, such as the hosted-mode browser provided by GWT.

To test native JavaScript code, GWT provides a subclass of JUnit's `TestCase`

method called `getModuleName`, which returns a string containing the name of your GWT code module as defined in your application's module configuration XML file. (GWT applications can be grouped into reusable modules, each of which contains an XML file descriptor with information including source code location, dependencies, target browsers, etc.)

When you run your test, the GWT framework starts up an invisible (or “headless”), hosted-mode browser and

```
public void setStylePrimaryName(int row, int column, String styleName) {
    UIObject.setStylePrimaryName(getCellElement(bodyElem, row, column),
                                styleName);
}

private native Element getCellElement(Element table, int row, int col) /*-{
    var out = table.rows[row].cells[col];
    return (out == null ? null : out);
} */;
```

Listing 1

```
public class MeetingSummaryLabelTest extends GWTTestCase {

    public String getModuleName() {
        return "com.danielwellman.booking.Booking";
    }

    // Add tests here
}
```

Listing 2

called `GWTTestCase`. This base class allows you to implement your JUnit test case as you normally would. In fact, `GWTTestCases` look almost identical to the standard JUnit `TestCase` shown in listing 2.

The only visible difference is that all `GWTTestCases` must override an abstract

then evaluates your test case against it. What this means is that all the facilities of the hosted browser are available to your test case. You can run native JavaScript functions, render widgets, or invoke asynchronous remote procedure calls. Furthermore, you can run your tests either as a hybrid of Java and JavaScript



code (“hosted mode”) or compile and run all your Java code as JavaScript (“Web mode”). The GWT team recommends that you run your tests both in hosted mode and Web mode, since there are a few subtle differences between Java and JavaScript [1], which could cause unexpected behavior.

Being able to test native JavaScript code in your Java JUnit tests is great, but there are some limitations. First, the normal browser-event mechanisms don’t work as expected in test mode. For example, you can’t programmatically click a button and expect the corresponding event handlers, such as `onClick()`, to fire. Selenium [2], the open source testing tool, can control a real browser and is a helpful alternative in this situation.

There are also performance considerations; the tests are slower than standard JUnit `TestCases`. Running a `GWTTestCase` forces a compilation of the source code in your module, which incurs an initial startup delay. Furthermore, each individual test method is wrapped by logic that starts up and shuts down the headless browser, which can take several seconds. Some testers would call these integration tests, not unit tests, since they involve other systems, cross language boundaries, and are slow to execute.

So when should you extend a standard JUnit `TestCase` vs. a `GWTTestCase`? In general, you should prefer standard JUnit `TestCases` because they run orders of magnitude faster than a `GWTTestCase`. If your code executes native JavaScript, however, or uses the libraries supplied with GWT, then your test must extend `GWTTestCase`. The upshot is that even if you simply instantiate a widget in the code being tested, you will have to test this using a `GWTTestCase`. You might try to find another design approach that avoids this native code requirement, such as moving the logic to another class.

## GUI Design Patterns

To build a testable GUI application, there are several design patterns and techniques you can use. All of them focus on one core principle: Move as much logic as possible out of the view and into other, more easily testable layers. One common pattern is known

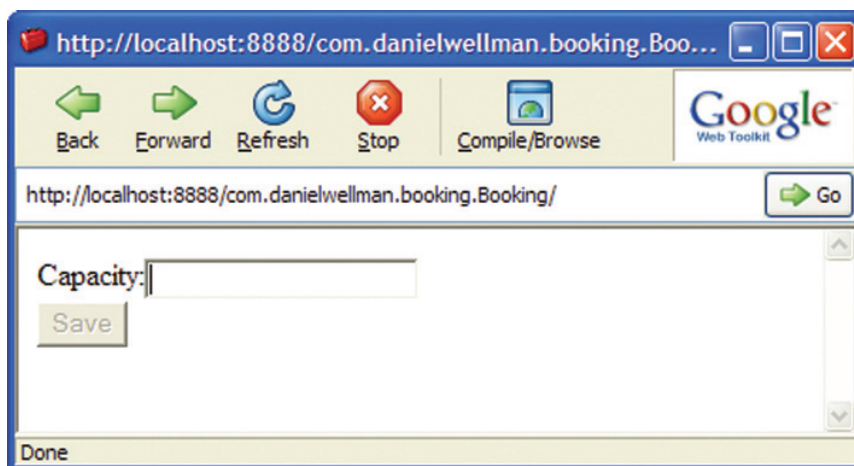


Figure 1: The first iteration of the UI for the booking application

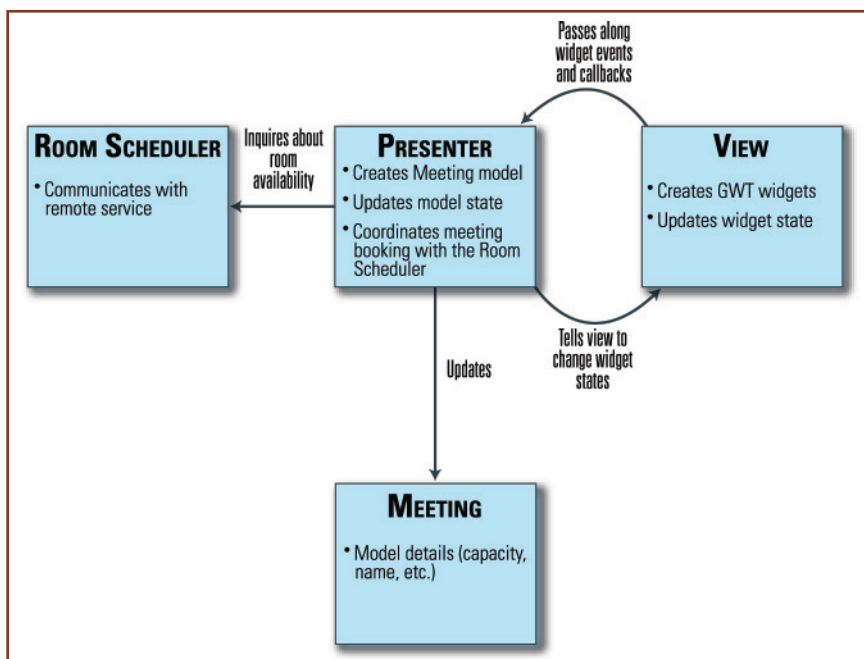


Figure 2: Object responsibilities and interactions for the booking application

as Model-View-Presenter, where a presenter object acts as a mediator between the view (GUI) and model objects and instructs the view layer to change states in response to user input or model changes. Martin Fowler has described a few variants of this pattern in his book [3], including Supervising Controller and Passive View. Both patterns push all logic- and event-handling code into the presenter, but they differ in how much the view knows about the model. Presenters hand the model objects directly to the view in the Supervising Controller pattern; then the view picks the appropriate information to display. In Passive View, the view layer knows nothing

about the model objects, and the presenter communicates model details to the view in terms of primitives, such as strings and numbers. Michael Feathers’s paper “The Humble Dialog” [4] provides a good introduction to the subject, and Martin Fowler’s book is a good resource for the other variations.

## Example

To illustrate some of these concepts, let’s take a look at building a small portion of an application. For this example, suppose we’re building an online application for booking meeting rooms at a conference center. A user will need to specify some details about the meeting,

```

@RunWith(JMock.class)
public class PresenterTest {

    Mockery context = new Mockery();

    @Test
    public void anUnavailableRoomDisablesTheSaveButton() {

        final MeetingView view = context.mock(MeetingView.class);
        final RoomScheduler scheduler = context.mock(RoomScheduler.class);

        final Meeting meeting = new Meeting();
        final Presenter presenter = new Presenter(meeting, view, scheduler);

        // The schedule service will reply with no available capacity
        context.checking(new Expectations() {
            {
                allowing(scheduler).canAcceptCapacityFor(meeting);
                will(returnValue(false));

                one(view).disableSaveButton();
            }
        });

        presenter.requiredCapacityChanged(new FakeTextContainer("225"));

        assertEquals("Should have updated the model's capacity", 225,
            meeting.getCapacity());
    }
}

```

### Listing 3

including the expected capacity and date. The application will check with a scheduling back-end service to determine if the room is available. If it's not available, the Save button will dim and a message will be displayed. See figure 1 for a sample layout of this dialog.

After some quick drawing at a whiteboard, we come up with a rough sketch of the objects involved, as shown in figure 2.

## Building the Presenter

The key to testing presenters is to keep in mind that they are plain old Java code and can be tested like any other Java code with JUnit. A mock-object library like JMock [5] can be used to test the interactions between the presenter and the view components.

Let's tackle a small slice of the following functionality: The user enters a meeting capacity that cannot be scheduled. First, the view will notify the pre-

senter that the user changed the value of the capacity text field. The presenter will then ask the RoomScheduler service if it can accept a new meeting with the specified capacity. Finally, the presenter will tell the view to disable the Save button. Listing 3 shows a test for this scenario.

This is an interaction-based test using JMock to provide test doubles for the MeetingView and the RoomScheduler. We stub out the scheduler to reply that it cannot accept the capacity for the meeting and expect our view to be told to disable the Save button. Note here that the view ends up being fairly dumb; it does nothing but notify the presenter whenever the required capacity is changed.

This code requires that we specify an interface for our view:

```

public interface MeetingView {
    void disableSaveButton();
}

```

and for our service:

```

public interface RoomScheduler
{
    boolean canAcceptCapacityFor(
        Meeting meeting);
}

```

The code that passes this test is fairly simple, as shown in listing 4.

The presenter is responsible for orchestrating the call to the remote service and instructing the view to disable the Save button. Note also that we're choosing to let the presenter maintain the state of the Meeting object, so that all UI events ultimately modify this object.

This is a very simple implementation, but it's far from the completed design. Our next test would probably check that setting an acceptable capacity enables the Save button and drives us to make either a new enableSaveButton method or a generalized setSaveButtonAvailable method on the view. We're still testing

```

public class Presenter {
    private Meeting meeting;
    private MeetingView meetingView;
    private RoomScheduler roomScheduler;

    public Presenter(Meeting meeting, MeetingView meetingView, RoomScheduler
roomScheduler) {
        this.meeting = meeting;
        this.meetingView = meetingView;
        this.roomScheduler = roomScheduler;
    }

    /**
     * Callback when the view's capacity text box changes
     *
     * @param textField the capacity TextBox widget
     */
    public void requiredCapacityChanged(HasText textField) {
        meeting.setCapacity(Integer.parseInt(textField.getText()));
        if (!roomScheduler.canAcceptCapacityFor(meeting)) {
            meetingView.disableSaveButton();
        }
    }

    protected Meeting getMeeting() {
        return meeting;
    }
}

```

**Listing 4**

```

package com.google.gwt.user.client.ui;

public interface HasText {

    /**
     * Gets this object's text.
     */
    String getText();

    /**
     * Sets this object's text.
     *
     * @param text the object's new text
     */
    void setText(String text);
}

```

**Listing 5**

```

public class FakeTextContainer implements HasText {
    private String text;

    public FakeTextContainer(String text) {
        this.text = text;
    }

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }
}

```

**Listing 6**

plain Java objects that don't require any JavaScript, so these tests run quickly.

Note the argument to `requiredCapacityChanged` is of the type `HasText`. This turns out to be an interface that is part of the GWT libraries, as shown in listing 5.

This simple interface is used by many

GWT components and allows manipulation of a widget's text contents, including the `TextBox` in our example. This interface is extremely useful for testing because we don't need to pass in a real `TextBox`. Thus, we avoid instantiating a text input in the DOM, requiring our test to extend `GWTTestCase` to run in

a real browser. In listing 6, I've made a simple, fake implementation that wraps a string.

And finally, the view implementation is shown in listing 7.

As you can see, there's not much logic here. Most of the code is involved in setting up the event listeners and config-



```

public class MeetingViewWidget extends Composite implements MeetingView {
    private Button saveButton = new Button("Save");
    private TextBox capacityText = new TextBox();

    public MeetingViewWidget() {
        VerticalPanel mainPanel = new VerticalPanel();

        HorizontalPanel row = new HorizontalPanel();
        row.add(new Label("Capacity:"));
        row.add(capacityText);

        mainPanel.add(row);
        mainPanel.add(saveButton);

        // Start with the save button disabled
        saveButton.setEnabled(false);

        // Here the view is responsible for creating the model and presenter
        final Presenter presenter = new Presenter(new Meeting(), this,
            new RemoteRoomScheduler());
        capacityText.addChangeListener(new ChangeListener() {
            public void onChange(Widget sender) {
                presenter.requiredCapacityChanged((HasText) sender);
            }
        });

        initWidget(mainPanel);
    }

    public void disableSaveButton() {
        saveButton.setEnabled(false);
    }
}

```

**Listing 7**

```

public AlternatePresenter(Meeting meeting, MeetingView meetingView,
    RoomScheduler roomScheduler) {
    this.meeting = meeting;
    this.meetingView = meetingView;
    this.roomScheduler = roomScheduler;
    // Register to receive all widget callbacks to this presenter
    meetingView.registerPresenter(this);
}

```

**Listing 8**

uring the display widgets. So how do we test it in a `GWTTestCase`?

We don't. In fact, there's not much here that can be tested in an automated test. As stated earlier, event propagation won't work by default in a `GWTTestCase`, and the layout of widgets is often best checked visually. If you are building a widget library, then you might want to write `GWTTestCases` that test the widget through its API, which is what

Google does with the widgets included in GWT, such as `Button`, `TextBox`, and `Tree`. However, these tests are slow (a sample widget test takes twelve seconds on my two-year-old workstation), and any complex logic could be moved into a simple presenter object, which could be tested in a plain old, fast `JUnit TestCase`. For more ideas for testing GWT widgets, see the *StickyNotes* for a link to my blog post on the subject.

Note here that the view is instantiating the model and presenter objects, which is one way of ensuring that the presenter is instantiated with a "live" view. You also could have some higher-level application object construct the view and pass it to the presenter, which would then need to register itself with the view so all the controls know where to send their events. This would look something like listing 8.

## Testing Asynchronous Access to Remote Services

GWT provides a remote procedure call (RPC) mechanism that enables passing Java objects between the server and client using a server-side serialization library. `GWTTestCase` supports testing of these features by providing utility methods that facilitate writing asynchronous tests. Most of the infor-

mation available on `GWTestCase` focuses on these RPC cases, and I recommend reading it for the full story. For a brief introduction, refer to the GWT documentation page titled "JUnit Integration" in the section "Asynchronous Testing" or, for a deeper example, review the book *GWT in Practice* [6].

## Reflecting on this Design Approach

My team used the design approach I've described in this article on a project and found it worked well. A disadvantage of this design is that it relies on the views' correctly registering the callback events with the presenter. Since this logic was almost too simple to break, we accepted these limitations on our project. For end-to-end integration tests, we used Selenium to control an instance of Firefox and Internet Explorer. These tests filled in the cracks to ensure we had widgets properly wired into their corresponding presenters.

Testing GWT applications, like testing Swing or other desktop client applications, can be fairly succinctly summarized as follows: *Don't put logic in your view components*. If you find complicated logic in your view, see if it can be moved into the model objects or the presenters. When you need to test view component behavior, JavaScript, or remote server communication, use a `GWTestCase`. If you have too many slow `GWTestCases`, see if there's some design change that doesn't require testing inside a real browser. Let the tests help guide your design, and you'll be on your way toward making Ajax development fun and relatively painless. **{end}**

### REFERENCES:

- [1] "Compatibility with the Java language and libraries" in the Google Web Toolkit online documentation.  
[tinyurl.com/6d4tg6](http://tinyurl.com/6d4tg6)
- [2] Selenium. [selenium.openqa.org/](http://selenium.openqa.org/)
- [3] Fowler, Martin. [martinfowler.com/eaDev/uiArchs.html](http://martinfowler.com/eaDev/uiArchs.html)

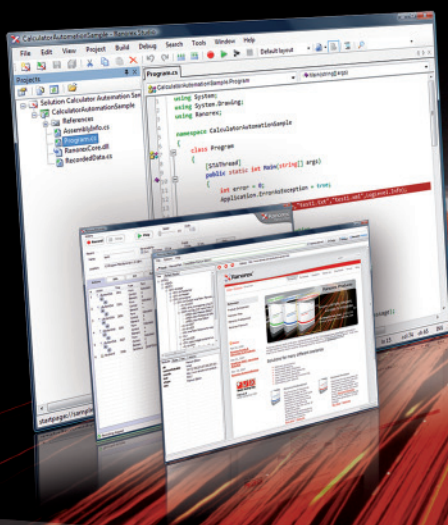
- [4] Feathers, Michael. "The Humble Dialog Box." Object Mentor, inc., 2002. [www.objectmentor.com/resources/articles/TheHumbleDialogBox.pdf](http://www.objectmentor.com/resources/articles/TheHumbleDialogBox.pdf)
- [5] JMock. [jmock.org/](http://jmock.org/)
- [6] Cooper, Robert and Collins, Charlie. *GWT in Practice*. Manning Publications, 2008.

### Sticky Notes

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

- Testing GWT widgets
- Infrastructure tips
- More information
- `GWTestCase` gotchas

# GUI Test Automation for Everyone



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

FREE  
TRIAL

Download - Learn more  
[www.ranorex.com/no-limits](http://www.ranorex.com/no-limits)

 **Ranorex®**

# AGILE SOFTWARE DEVELOPMENT TRAINING

*Accelerate Your Career & Empower Your Team*



## BUILD-YOUR-OWN TRAINING WEEK

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



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

## AGILE SOFTWARE DEVELOPMENT TRAINING WEEK LOCATIONS

March 30–April 3, 2009

April 20–24, 2009

May 11–15, 2009

Washington, DC

San Francisco, CA

Chicago, IL

## Choose from 4 specialized training courses:

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

- Scrum Master Certification
- Practical Test-Driven Development

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

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



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



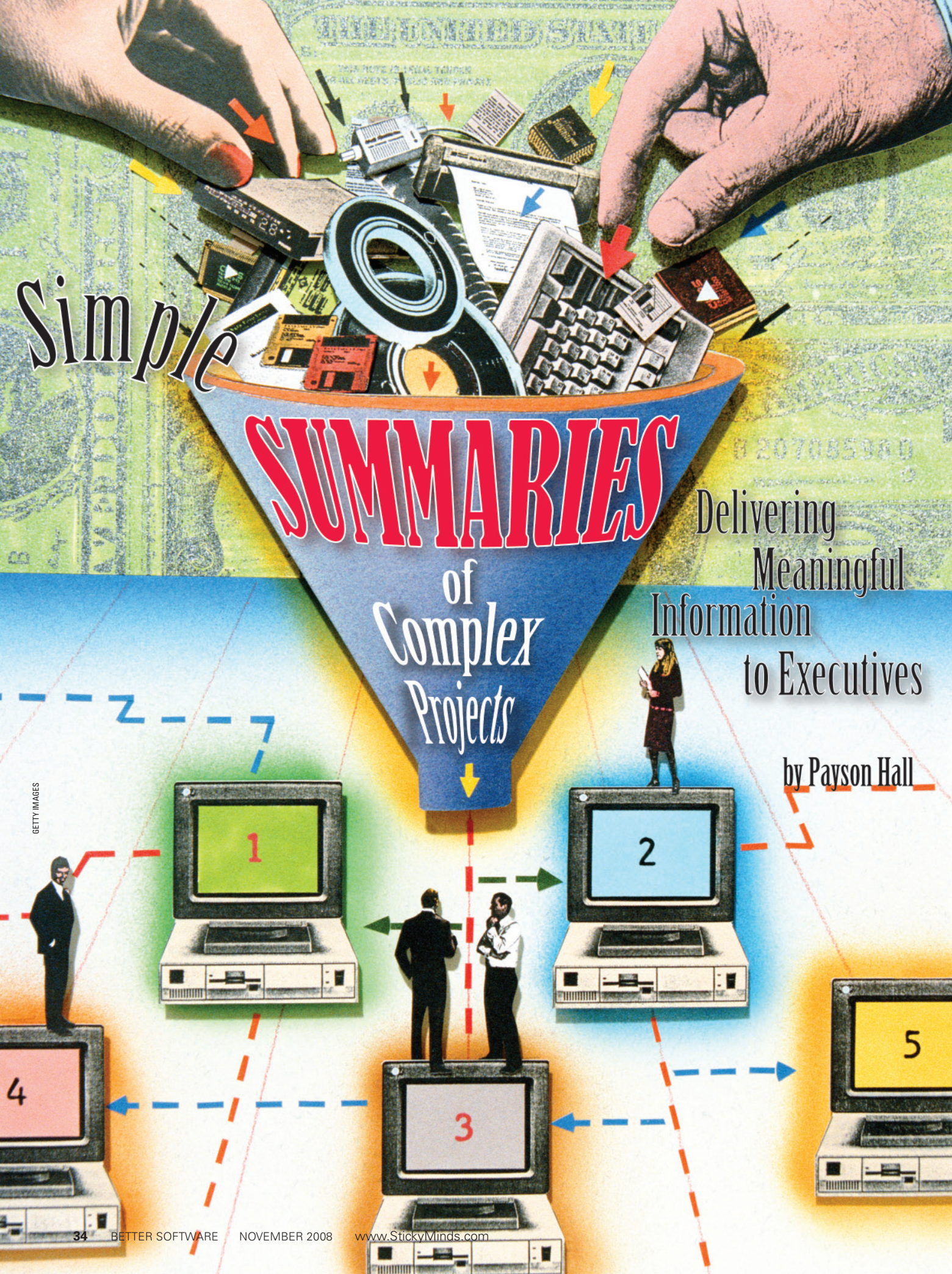
Simple

# SUMMARIES

of  
Complex  
Projects

Delivering  
Meaningful  
Information  
to Executives

by Payson Hall



GETTY IMAGES



**T**he boss says, “I need a status briefing on the Alpha project tomorrow morning. It needs to fit on six to ten PowerPoint slides, and you will have twenty minutes to present it.”

A request to distill complex project status into a twenty-minute briefing might sound like something from Dilbert’s pointy-haired boss, but if your organization has dozens of projects active in its portfolio, this may be a common, necessary, and legitimate request.

How can we meaningfully summarize—in a brief status report without losing important details—the successes and setbacks projects experience? The short answer is, we can’t—some information will be omitted. Thus, it is vital to select and present the information needed to support informed decision making.

## Is Status Reporting Necessary?

Many see project status reports as little more than a ritual sacrifice of trees made to appease the pagan gods (pointy-haired executives). But, here is a secret: The leaders in some organizations actually use status information to support decisions and help them steer.

Imagine that your organization manages a multi-million-dollar portfolio of projects. Some projects are simple, inexpensive, low risk, and short; others are complex, expensive, mission critical, and may extend for years. An organization’s executives must routinely make decisions about these projects, including:

- Which projects are going well? Which need special attention?
- Which projects will continue as defined, which will be cancelled, and which will be accelerated, slowed, or postponed?
- Which budgets will be cut? Which will be augmented?
- Which projects are not on track to meet expectations?
- How should the organization respond to that?
- What is the relative priority among the projects?
- Which projects are expected to deliver the most value to the organization in the near future?

It’s obvious that these decisions can be difficult, complex, and easy to get wrong. What decision makers need is timely and accurate status information summarized by knowledgeable and trustworthy people who are close to the effort and know what is important.

We sometimes imagine our leaders (managers, executives, legislators, generals) as super-beings who track all details of the universe simultaneously, using their massive brains and uncanny intuition to sort it all out with clarity. In reality, most executives I’ve met are clever, but thus far none has revealed any super powers to me. At the risk of heresy, some executives are probably no smarter or experienced than you. Scary, isn’t it? Perhaps you were hoping your organization had adult supervision? Don’t panic! This just means the decision makers in your organization are mere mortals who need your help to deal with the complexity of their projects.

## What Information Defines a Project?

A project is a temporary effort undertaken to accomplish a specific goal within defined resource bounds. Examples of projects:

- Obtain suitable manufacturing and office facilities in Oklahoma City for 300 staff to occupy by January 1, 2010, for \$5 million.
- Design, build, and test Version 2.0 of the WhizBangWidget product for \$2 million with release to manufacturing to occur by June 1, 2009.
- Procure, install, customize, and roll into organization-wide production a suitable replacement for the XYZ software system by December 31, 2010, with a budget of \$15 million.

Projects exist in three dimensions: *scope* (what you want), *schedule* (when you want it), and *resources* (what you are willing to invest to get it). Most project management methodologies and processes can be mapped to the three-step model shown in table 1.

Larger projects may adopt a divide-and-conquer strategy that partitions an effort into multiple smaller projects that are done individually. The important point is that you cannot have an intelligent discussion about a project without addressing three project dimensions (scope, schedule, and resources).

Step 1: Define	Determine the boundaries of the effort (scope, schedule, resources).
Step 2: Plan	<p>Determine the work (scope) required to complete the project, figure out when it will happen (schedule), decide who will do it and what it will cost (resources), and compare this to the definition from step 1.</p> <ul style="list-style-type: none"> <li>• If the solution from step 2 fits within the boundaries from step 1, proceed to step 3.</li> <li>• If the solution from step 2 doesn't fit, tinker with the solution or your problem definition to see if you can make it fit (this is called optimization and renegotiation of the definition). If they fit, go on to step 3, else stop (project is not feasible).</li> </ul>
Step 3: Do work and monitor the effort	<p>Do some increment of the defined work and monitor your progress:</p> <ul style="list-style-type: none"> <li>• Scope—Did you do what you thought you were going to do?</li> <li>• Schedule—Did it happen when you thought it would happen?</li> <li>• Resources—Did it cost what you thought it would cost (people, money, etc.)?</li> </ul> <p>Compare the results to your plans and adjust from there.</p> <ul style="list-style-type: none"> <li>• If your plans seem consistent with reality to date, do more work and continue monitoring your progress.</li> <li>• If your plans are wrong, rework your plans (go back to step 2) or try to adjust your approach to get back on track.</li> <li>• If the project is over, celebrate.</li> </ul>

Table 1

Pop Quiz: Identify three project dimensions that should be represented in any project definition, project plan, or project status report:

- Rock, paper, scissors
- Larry, Moe, curly
- Sun, moon, stars
- Scope, schedule, resources
- I like pie

The correct answer is D (even if you do like pie).

## Scope, Schedule, and Resources: How Hard Can That Be?

Describing status in terms of scope, schedule, and resources sounds straightforward, but it can be challenging to determine our status and to explain it to others. In the future, when our presenta-

tion systems do a better job of projecting three-dimensional animated images, this task may be easier. But here in the early twenty-first century, we often make do with independent discussions of scope, schedule, and resources and struggle to represent the interrelationships among these factors.

For example, I recently attended a project meeting in which the project manager presented status suggesting his project was spending significantly less than expected to date but was running two months behind schedule. The busy executives at the meeting considered this briefly and inferred that the project was being schedule delayed but was going to finish under budget. That was one way to interpret the data, but that interpretation missed the connection between schedule and resources.

Project start had been delayed, so the resources that had been budgeted for the first months of the project were not expended. The project had anticipated needing sixty person-months of effort in the first six months. Because of the delayed start, only forty person-months had been consumed, but that was because some of work scheduled for the first six months *had not yet been performed*. The cost estimate to complete the project had not changed; the project just hadn't spent as much to date as expected.

The challenge is to accurately reflect project status while representing the interrelationships among scope, schedule, and resources. If you think of plans as maps that describe the route to the project objectives, project performance can be described as consistency with

	Total		August	September	October	November
Estimates	\$200,000		\$52,000	\$45,000	\$48,000	\$55,000

Table 2



or variance from current plans. This requires up-front planning from the project manager to describe the work approach and ongoing effort to track progress against those plans. It also requires effort on the part of the people receiving status to take the time to understand the meaning and implications of status. Let's examine a hypothetical project beginning with resources and working through to scope.

## Resources

Pictures can show how the budget (resource) plan and progress against it are related, as shown in table 2. Imagine a four-month project expected to consume \$200,000. We arrived at this figure from estimates of labor as well as hardware and software purchasing allocated across the life of the project.

When the project is approved, these estimates become the "baseline" plan, the basis for future comparisons of progress and performance.

Let's create a status report for this project in late September. Actual expense information (hours spent and hardware and software purchased) is known for the month of August and for the first half of September.

First, let's focus on what we know about September. We had originally budgeted \$45,000 for September, as shown in figure 1. As of September 15, we had spent only \$20,000; however, we anticipate spending an additional \$40,000 this month. Yikes! This information says we expect to be \$15,000 (33 percent) over budget in September. That is the danger of looking at a point in time out of context. Figure 1 shows what we believe is happening in September, but it doesn't explain why or provide much context for the information.

What might be helpful would be to show September in the context of the entire project budget, as shown in figure 2.

We now see that in September we expect to spend \$15,000 more than initially planned, but October is now expected to cost \$12,000 less than originally planned. The rationale for the expense shift (or whatever is going on) should be presented in the status report, but this bigger picture supports that explanation. It might be something as simple as the

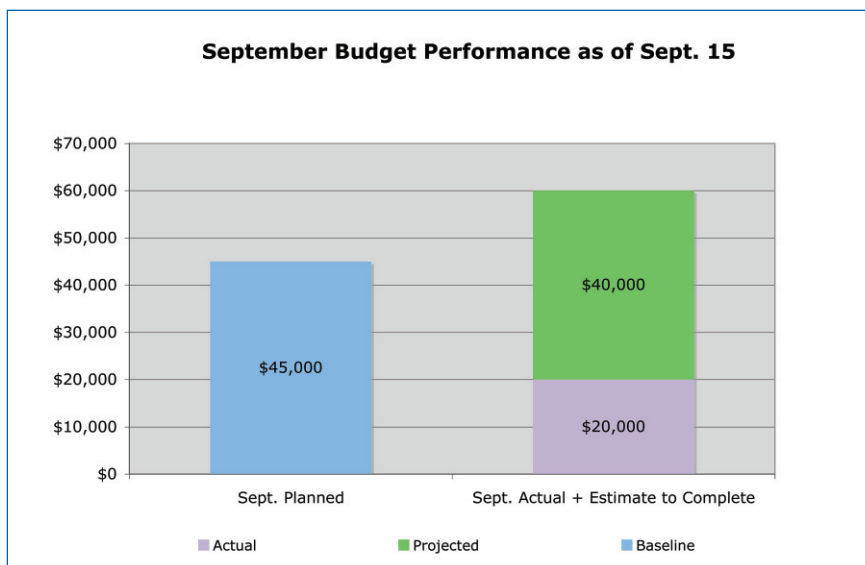


Figure 1

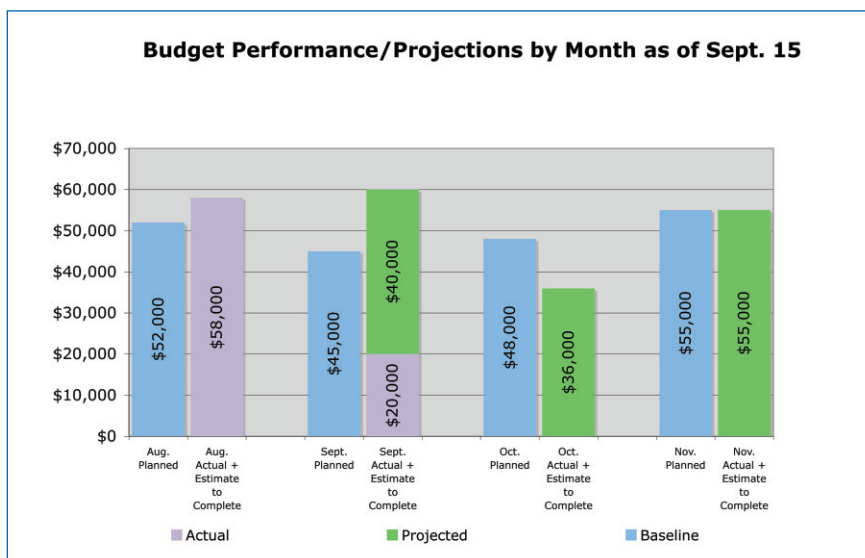


Figure 2

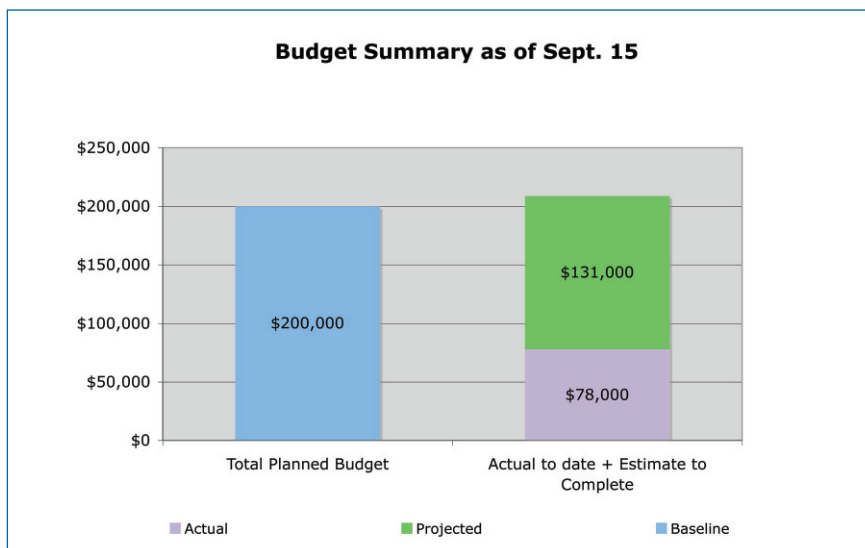


Figure 3

Milestones summarize schedule information. They do not represent effort, but instead represent a moment in time. Milestones have completion criteria, but the effort required to do the work to meet the criteria is assigned to tasks.

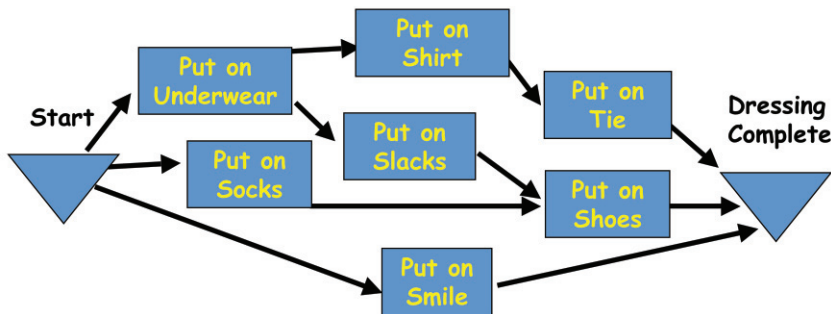


Figure 4

hardware initially planned for October purchase is being procured early. The resource consumption (aka “burn”) rate often varies a bit from one month to the next, so small adjustments aren’t necessarily a concern. To track overall budget performance, the missing piece of the puzzle is a summary of all expenses and updated estimates compared to the original baseline budget, which might look like figure 3.

As shown in figure 3, the current expectation is that the project will finish about \$9,000 over budget. While this might be an issue for the executive sponsors, it helps put September’s progress in context.

While budget information alone will not help us know whether the project is on track to deliver a quality product at the desired time, it does represent an important part of the overall picture.

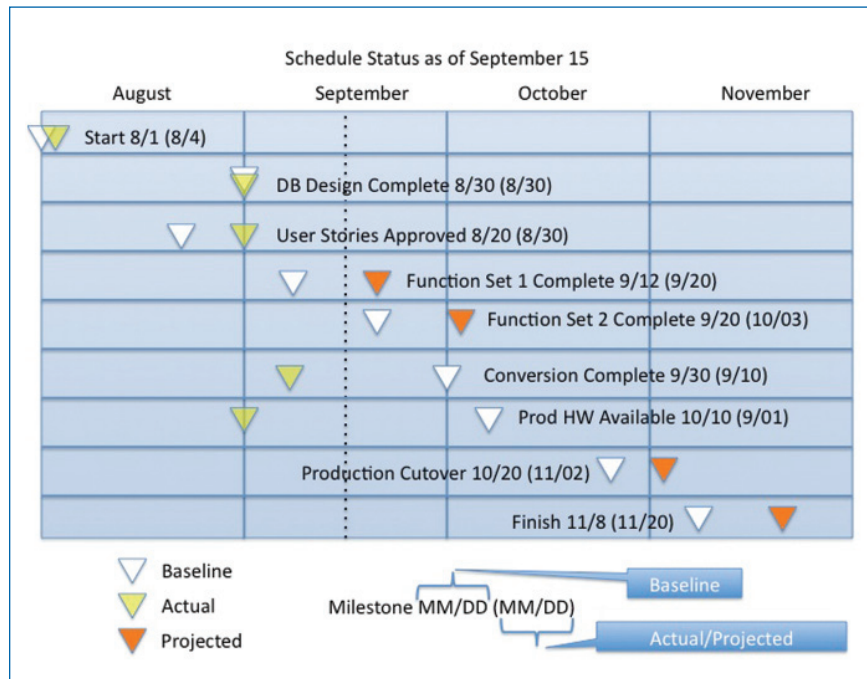


Figure 5

## Schedule

How long will the project take? Are we on track to deliver the product as promised? Many organizations ask these questions, but few project managers have ready access to the information needed to answer them.

Good schedules are based upon defined chunks of project work called tasks, assumptions about the effort required to perform these tasks, and assumptions about the availability of people to work on the tasks. This information is combined to create duration estimates for each task. Task duration combines with task sequence information to create baseline schedule information.

The schedule doesn’t just answer the question “When will the project be finished?” The schedule also describes a baseline that indicates when the project’s component tasks will occur. Like resource status, the challenge when providing schedule status is to summarize information so that executives don’t get buried in data and lose important information.

Anyone who has worked on a large system project has seen complex schedules with hundreds of tasks maintained in a scheduling tool. If it is kept current, this detailed information can be useful to a project manager; in its raw form it is usually not helpful for briefing executives. Fortunately, complex schedule information can be summarized with *milestones*.

“Milestone” is a technical scheduling term that is worth understanding because milestones are useful. A milestone represents the moment in time when a specified condition is met. People often confuse tasks with milestones because the two are related, but they are as distinctly different as programs and data. A task describes a unit of work. Tasks consume resources and have a duration that describes how much time is estimated to elapse between the start and finish of the task. Milestones consume no resources, have zero duration (they represent a specific moment in time), and are used to give visibility to the completion of one or more tasks or to represent significant moments in the project’s life. Figure 4 provides an example of a milestone.

The milestone labeled “Dressing

**“When asked whether the project manager should try to show bad news to executives in the best possible light, a friend once answered, ‘I didn’t invent reality; they pay me to explain it to them.’”**

Complete” is used to designate the completion of the tasks that precede and comprise it. The milestone is defined as “This milestone is complete when ‘Put on Tie,’ ‘Put on Shoes,’ and ‘Put on Smile’ are all complete.” If you are in my carpool, you don’t care about the details of my morning dressing ritual, but you do care when I’m dressed and ready to go. This milestone encapsulates the details. I can tell you that I expect the milestone to be complete at 7:30, and you need not concern yourself with when I put on my socks.

Another use of milestones is to identify and bring visibility to a project’s external dependencies—key events in the project that are not directly within our control. For example, imagine that operations tells us that our server will be available for use on August 15. One way to show that event in our schedule is by creating a milestone called “New Server Available to the Project Team.” The milestone would be scheduled to occur on August 15 based upon the information from operations, and we would plan our project as if this were the case.

Examples of milestones include:

- Unit testing for all modules complete
- Customer approval of prototype received
- Data conversion complete
- New disk space available to the project team
- Project ready for Beta distribution

When the initial project schedule is constructed, prudent project managers identify milestones that are significant and can be explained easily to executives. The baseline schedule shows when these milestones are expected to occur. Subsequent status can show any changes in that expectation.

The milestone chart in figure 5 shows

schedule performance for our sample project.

What is not shown in this summary but must be carefully monitored and managed are the relationships and dependencies among the milestones. Failure to complete some milestones on time can have a ripple effect through the project that may require explanation. Project management tools help manage these dependencies.

## Scope

The scope dimension of the project accounts for whether we are accomplishing the objectives initially envisioned for the project. This includes implementation and delivery of the products or services with all features, functions, performance, reliability, quality, and compliance with process constraints that were agreed upon. Measures of scope can be complicated and vary by project, but a good project manager seeks useful summaries and surrogates for the dimensions of quality important to the project and

its sponsoring executives and represents these as part of status.

Again, the important idea is to capture the baseline (original scope goals), the progress toward the baseline, and any changes anticipated.

Imagine we begin a software project with the assumption that the final product will represent fifty user stories. Using fifty stories as a crude baseline measure for scope, we could track stories through a “lifecycle” consisting of:

- Identification—Analyst identifies story
- Approval—User agrees story correct and in scope
- Implementation—Add support for the story to the product
- Testing—Complete testing for the story
- Acceptance—User signs off that the story is acceptable as implemented

Figure 6 shows not only status for the date in question but also progress

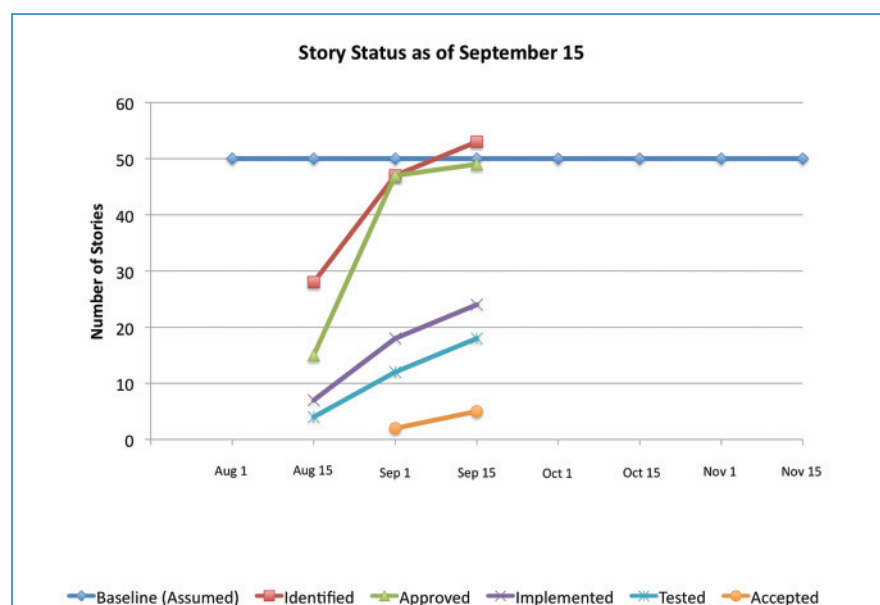


Figure 6



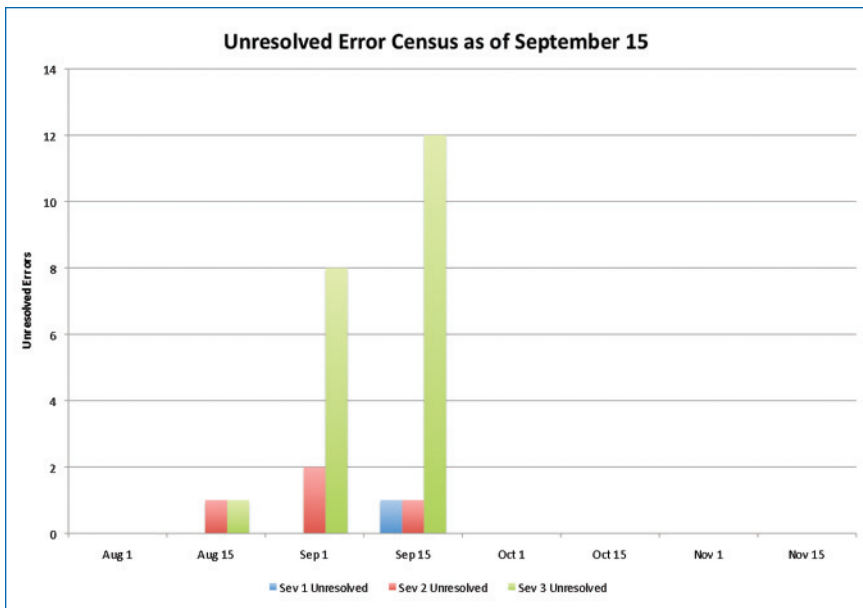


Figure 7

toward the baseline over time. If more than fifty stories are required to implement essential functionality, this chart graphically shows that the original assumptions about scope were incorrect.

Identified and unresolved faults in the product also can be a useful way for executives to see progress toward the goal and get a sense of product quality. Assume we have an agreed-upon method for sorting faults into severity 1 (fatal), severity 2 (functional), and severity 3 (cosmetic) errors. Displaying error counts alone may not be helpful. If we track the number of open errors by severity over time, as shown in figure 7, it may be easier to explain product status.

We would hope that the unresolved error count would peak and then decline as we near the end of the project. We also might expect that severity 1 and 2 errors would be a priority and that this would be reflected in trend data. Where either of these expectations is not the case, this should be brought to the attention of the sponsoring executives and discussed as part of status.

## The Rest of the Story: Issues, Decisions, Changes, and Risks

Establishing baseline expectations for scope, schedule, and resource performance occurs initially as part of definition and planning. If project goals don't

change and work proceeds according to plan without significant issues or risks, then simply describing scope, schedule, and resource performance against initial baselines will be sufficient status. Unfortunately, this is *never* the case.

Describing progress against baseline expectations in the three dimensions described previously is necessary but not sufficient. Executives need to know not only what has happened but also what is happening and why. Performance against baseline expectations sets a context for the discussion. Other essential parts of project status include:

- Significant project *issues* (surprises, incorrect assumptions, challenges, and any points of dispute or disagreement to be elevated to the executive level)
- Any *decisions* that must be made or supported at an executive level
- Significant *changes* to the project approach or definition
- New, changed, and significant *risks* to project success

All of this information must be monitored, assessed, packaged, and presented on a regular basis to allow informed, executive decision making. If this sounds like work, you are correct. Establishing and maintaining the systems and processes to do this are the responsibility of the project manager.

What might a real-world status report look like? While the content and format will vary by project and organization, a sample status report can be found in the StickyNotes.

## Parting Thoughts

Effective status reporting is about efficient information transfer from the project manager to the project's sponsoring executives. When asked whether the project manager should try to show bad news to executives in the best possible light, a friend once answered, "I didn't invent reality; they pay me to explain it to them." Wise words.

Monitoring complex aspects of project performance and then succinctly and effectively communicating project status to executives can be challenging. The goal is not the ritual sacrifice of trees or painting a rosy picture to assure that your project continues to receive management support. The intent of effective status reporting is to provide executives with the timely information they need to do *their* job of steering the organization, establishing and supporting priorities, and allocating resources.

To learn more, show a sample status report to your executives and ask them what information they wouldn't care to know about; their answers should be instructive. Create the status reports that you would want to receive if you were an executive responsible for multiple large organizational investments and had only a few hours per month to stay on top of progress.

Keep a record of your status reports, and you will find that they provide an excellent audit trail and raw material for retrospectives that can lead to process improvement of your project-reporting skills and your executive's sponsorship skills. {end}

### Sticky Notes

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

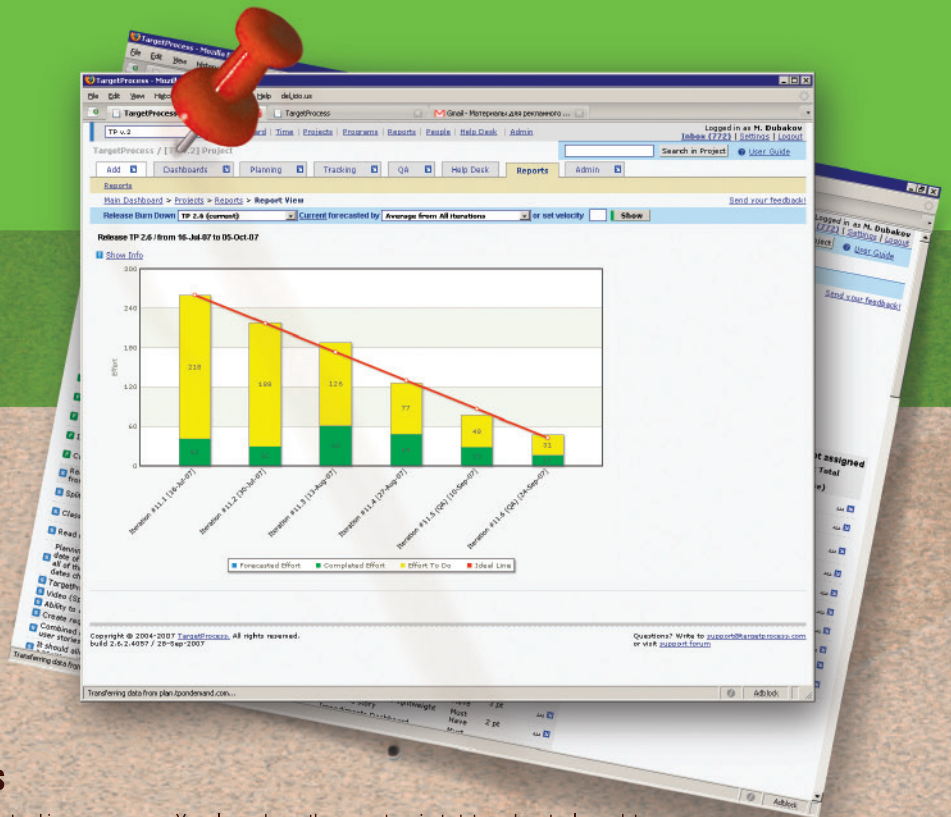
- Sample status report

*TargetProcess supports SCRUM, XP and other agile processes*

# TargetProcess v.2

Agile Project Management & Life-Cycle Software

*simplicity in mind*



## Benefits

Real time progress tracking	You always know the current project state and next release date
Full support of iterative development	Use it for projects of all sizes
Responsive user-friendly interface	The learning curve is smooth and straightforward
Variety of productivity tools	Execute common tasks faster
Highly customizable	Quickly adapt TargetProcess to your development process

**Key Features** *TargetProcess helps software development companies reduce the complexity of software project management.*

- Full Iterative Development support
- Integrated bug tracking, integrated test cases management
- Highly customizable dashboards
- Release and iteration planning via drag & drop
- Development process customization
- Subversion integration
- Real time progress tracking
- Bug Submission Tool (Tp.Tray)
- Customizable workflows
- Integrated time tracking; weekly time sheet
- Combined printable reports
- People allocations management
- Powerful Web Services API
- Integrated Help Desk

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

**tp v.2**



## Product Announcements

### IBM Jazz

AUSTIN, TX—Surgient has announced it will offer integrated solutions with products built using IBM Jazz collaborative technologies. IBM quality assurance users can use Surgient's Virtual Automation Platform to allocate IT resources and set up and tear down virtual labs on demand for software testing and quality assurance, introducing new efficiencies to the application development lifecycle.

By implementing the Surgient Virtual Automation Platform, development and QA organizations will benefit from Surgient's self-service IT solution for virtual lab management. IBM Rational customers using IBM Rational Quality Manager can request and build a live application testing environment with specific configurations without manual administration from a company's IT department, freeing up IT personnel to address additional business critical initiatives.

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

### Coverity Software Readiness Manager

SAN FRANCISCO, CA—Coverity, Inc. has announced the availability of Coverity Software Readiness Manager for Java. This product allows development managers, release managers, and executives to assess objectively the release readiness of their critical code by combining essential data from multiple sources including Prevent, Coverity's industry-leading static analysis product. Software Readiness Manager helps development teams deliver high-integrity code that successfully meets quality standards to align product development with business goals. Key features include:

- Automatic identification of failure-prone (high-risk) code across large and complex software systems
- Translation of large amounts of data from multiple tools into actionable, prioritized recommendations for improving code
- Correlation of test coverage data with high-risk code to determine if failure-prone areas are being sufficiently tested

- Elimination of issues due to poor coding practices earlier in the software development lifecycle, before they negatively impact development
- Creation of quality and risk benchmarks to identify code appropriate for reuse

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

### SI Tested

WILMINGTON, MA—Security Innovation has unveiled its SI Tested program. This new program offers a means of validating a company's security testing efforts in the absence of either industry or internal company standards for software security. This validation allows companies whose software programs undergo Security Innovation's security testing to display the SI Tested logo on their Web sites, marketing collateral, and other appropriate software packaging.

The SI Tested program is currently available to all companies that wish to establish stringent security standards

## Better Software Has Gone Digital!

*R*ead your favorite magazine straight from your desktop before it hits the street—check out the digital edition of this issue online today at [www.StickyMinds.com/September](http://www.StickyMinds.com/September)

Want to switch to the digital edition? Contact our subscription department at 1-800-450-7854 or [info@BetterSoftware.com](mailto:info@BetterSoftware.com) to find out how.





practices and instill customers with confidence in their products and practices. By proactively demonstrating their commitment and accountability to these security best practices through an independent third party, organizations that bear the SI Tested logo differentiate themselves and gain a valuable marketing asset.

In order to display the logo, Security Innovation must evaluate a company's software. The software must be subjected to a rigorous security testing process that identifies high-severity vulnerabilities. The testing process ensures that critical vulnerabilities are found in order to protect customers and the security of their data.

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

### Test Automation Services

BURLINGTON, MA—Foliage, a technology consulting and product development company, has announced its Test Automation Services program, a package of services designed to address the business challenges faced by companies regarding the verification phase of

the software product development life-cycle.

- The **Strategy Formulation** service includes the creation of a product test automation strategy, encompassing planning, tool selection, technical approach, and return on investment (ROI) statement.
- The **Assessment** service provides a review of a client's existing test processes and offers ROI-based automation recommendations.
- The **Implementation** service consists of the implementation of the appropriate test automation capability along with integration into existing product development processes.

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

### Perforce 2008.1

ALAMEDA, CA—Perforce Software has announced the availability of Perforce 2008.1, the newest release of the Perforce Software Configuration Management (SCM) System that versions and

manages source code and all digital assets. With this release, Perforce SCM extends visual differencing functionality to images, enabling enterprise developers and engineers to manage all content with one SCM solution.

Perforce 2008.1 improves support for remote users. Enhancements to the Perforce Visual Client improve remote access to the Perforce Server by allowing users to work offline and to browse local files. To further speed remote access, the Perforce Proxy, a self-maintaining proxy server and the cornerstone of Perforce's distributed development solution, off-loads file decompression to the client.

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

### ANTS Profiler 4

CAMBRIDGE, UK—Red Gate has introduced ANTS Profiler 4, marking a major transformation in performance profiling, providing real-time analysis with an interactive timeline that gives developers a complete picture of performance within any region of code.

ANTS Profiler 4 enables developers

An offering of the  
School of Systems and Enterprises at  
Stevens Institute of Technology

# Systems-Centric Software Engineering

**Graduate programs tailored to the real-world education needs of today's Software and Systems Professionals**

Integrating the principles of Systems and Software Engineering to provide students with the skills required to conceive, develop, assure, and maintain complex software-intensive systems.

Graduate Certificate and Master's Degree programs are offered in convenient, flexible delivery formats.

Courses held on-site at corporate and government facilities, online via Stevens award-winning WebCampus, and on-campus in Hoboken, NJ.

To learn more about Stevens Systems-Centric Software Engineering programs and degree options, visit:

[www.stevens.edu/sse](http://www.stevens.edu/sse)



to profile .NET applications in one step. Simply enter settings (application to profile, profiling mode, timing display, etc.) and click the Start Profiling button. A new engine delivers eight times the performance of its predecessor, making it the fastest .NET code profiler with line-level timing.

ANTS Profiler 4 marks the end of the static performance snapshot. The new timeline feature enables developers to analyze results between any two points in real time. Click and drag over a region of activity on the timeline, and the tree view below automatically expands to show the worst-performing code within the region.

Visit [www.red-gate.com](http://www.red-gate.com) for additional information.

### ProjectLocker Now Integrates with Morph AppSpace

ATLANTA, GA—ProjectLocker has announced an integration with Morph AppSpace to enable developers to easily deploy applications directly to the cloud.

The integration enables ProjectLocker users to quickly deploy their applications to Morph's high-quality production environment directly from within the ProjectLocker user portal, without the hassles of procuring, configuring, and managing the Web application environment. This integration now allows ProjectLocker to expand its service offering to encompass the deployment and management phase of the software development lifecycle. By providing both a managed development environment along with a managed deployment environment, software teams can focus on using best practices to build their software without the expense in resources and man hours of building and maintaining their own solutions.

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

### CollabNet Releases CUBiT 2.0

CollabNet has announced the release of CUBiT 2.0, a Web-based tool that lets distributed development teams access on-demand build and test services.

CUBiT 2.0 addresses the process of

configuring servers for build and test by managing those configurations as “profiles” across the application lifecycle. Applying the cloud computing model to distributed development, CUBiT 2.0 lets teams access on-demand servers from private corporate data centers or public clouds, reducing development cycles and hardware expenses.

With CUBiT 2.0, teams can group and manage their computing resources as clouds. Clouds in CUBiT are groups of server pools from a corporate data center or from public clouds like Amazon EC2. Other new features in CUBiT 2.0 include support for LDAP/Active Directory, and advanced accounting and chargeback capabilities tied to role-based access control for allocating costs per server and profile type. A free trial of CUBiT 2.0 is available at [www.collab.net/products/CUBiT/tryit.html](http://www.collab.net/products/CUBiT/tryit.html).

# Passion. Improvement. Delivery.™



## Agile2009

Johanna Rothman, Chair  
**CHICAGO**

**Aug. 24 – 27**

[agile2009.agilealliance.org](http://agile2009.agilealliance.org)

Agile  
Alliance





# StickyMinds.com

*Map out your project success  
using StickyMinds.com as  
your resource guide*

#### **Benefits of joining StickyMinds.com:**

- Access daily news articles geared to our industry
- Have any of our six eNewsletters delivered straight to your inbox
- Enjoy Podcasts and Videocasts featuring industry experts
- Find software solutions in the comprehensive Tools Guide
- Search and read book reviews
- Post a question to your colleagues on our Discussion Board
- Find the resources you need for building better software
- Submit articles or technical papers for others to download

StickyMinds.com is the Web's first and most popular interactive community exclusively engaged in improving software quality throughout software development.

**Membership is free, so sign up today and start your journey to building better software!**  
[www.StickyMinds.com/join](http://www.StickyMinds.com/join)

# StickyMinds.com



# Things You Might Not Know About

## Using Testing Standards

by Claire Lohr

- 1 **THERE IS A BRAND NEW VERSION OF THE SOFTWARE TESTING DOCUMENTATION STANDARD.** IEEE Std-829-2008 Standard for Software and System Test Documentation was just published in July! It delineates a process for choosing and tailoring the testing documentation for a given project. It has updated the old, tried-and-true, test-related documents, and the new documents include a master test plan, documentation for each test level, a level interim test status report, and a master test report.
- 2 **THERE IS NO ONE SIZE FITS ALL.** Actually, there never was, and many long-suffering testers have tales of woe about endless, unnecessary documentation requirements. All of the new standards recognize this reality and recommend choosing useful contents and documents instead of trying to mandate everything for all projects.
- 3 **THE BEST EXAMPLE IS A REAL EXAMPLE.** Blank templates are an essential part of disseminating best practices and consistency throughout an organization. But, somehow, users of the templates seem to understand them much better from a real example. Note that the word “good” is missing. Even an irrelevant or somewhat mediocre example is better than no example at all.
- 4 **STANDARDS ARE CHEAPER BY THE DOZEN.** Actually, by the two to three dozen. The Institute of Electrical and Electronic Engineers (IEEE) has a software and systems engineering standards collection that can be purchased for about the price of three individual standards. I recommend getting an online license, so draft (why wait until final publication?) and newly published standards are automatically delivered as they become available.
- 5 **THE NATIONAL INSTITUTE FOR STANDARDS AND TECHNOLOGY HAS TWO FREE SECURITY TESTING STANDARDS.** From the [nist.gov](http://nist.gov) home page, click the Computer Security Resource Center link, then click the Special Publications link, and download the SP 800-115 Draft Guide to Security Testing and the SP 800-84 Guide to Test, Training, and Exercise Programs for IT Plans and Capabilities.
- 6 **THE BEST TESTING-RELATED DICTIONARY IS FREE AND REFERENCES MANY STANDARDS THAT ARE NOT FREE.** Go to [astqb.org](http://astqb.org) and click the Software Testing Glossary link.
- 7 **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO) AND THE IEEE ARE COLLABORATING TO AVOID DUPLICATION.** There is an agreement between ISO and the IEEE that allows ISO to have full use of designated, existing IEEE standards to develop the next generation of standards. It is possible to buy from the IEEE many ISO standards (including in collections), as they have been approved by both organizations.
- 8 **THE NEXT GENERATION OF TESTING STANDARDS WILL BE FROM ISO.** The IEEE, ISO, and the British Computer Society have turned all their testing-related standards over to ISO groups who are working to develop a collection of testing-related standards that keeps existing useful material and evolves the next generation of test standards.
- 9 **THERE IS A WAY TO GET A FREE COPY OF A NEW STANDARD.** Volunteer to participate in a working group developing a new standard, and you will get a free copy of it when it is published. Attendance at meetings is desirable but not required; however, some kind of participation is required.
- 10 **ANYONE CAN PROPOSE AN ENTIRELY NEW STANDARD.** That does not mean that it will be approved, but the instructions for how to do this are on the IEEE Web site.

# Metrics that Motivate

by Linda Hayes

It's said you can't manage what you can't measure. It's true that you need some type of measurement system to institute a meaningful incentive system for the people you manage, but to select the metrics that encourage the behaviors you need and reward the results you want, you first have to decide *what* you need and want. When it comes to testing, this is not as easy as it might sound.

If you want testers to find bugs, then counting bugs might seem like a simple system for tracking results. Unfortunately, this might encourage testers to focus on the fringes of functionality, engaging in bizarre behavior to uncover obscure defects and ignoring mainstream activities. It also invites derision from developers who view testers as adversaries who waste time trying to make them look bad.

You could count tests executed, but this could lead to high volumes of low-value tests, consuming time and resources without producing useful results. All tests are not created equal, and this approach might not distinguish basic menu navigation from more complex and critical end-to-end scenarios.

A better measurement might be the number of post-release bugs. This has the apparent benefit of putting focus on the issues that are important to users. The downside, though, is that testers rarely get to decide when the software is ready for release, nor do they always get all the resources they need. It may not be fair to hold the testers accountable for the ultimate outcome.

So what are the metrics that motivate? Put simply, they are the same metrics on which other managers and departments are measured: performance against plan.

## Plan to Succeed

To be measured against a plan, first you have to have one. A good test plan covers all aspects—what you are going to test and what you aren't, how long it will

take, what resources will be used, and what your assumptions are. Be sure to articulate the business goals for the product on which you are working, identify the potential risks of failure, and describe how you plan to avoid or mitigate them.

To be rewarded for meeting your plan, you have to get buy-in. This is critical. Meet with your users or other business representatives and review the plan with them.

Insist that they understand and approve it; don't accept perfunctory sign-off. Ask open-ended questions such as "What are we missing? What is most important to you? What is the worst-case scenario?"

Make sure the users grasp how the plan was designed to meet their goals, as well as all the factors that went into it. Once you have the plan approved, develop metrics that measure performance against it. This should take into account multiple aspects, such as coverage achieved, schedules met, and resource productivity.

## Negotiate the Reward

Next, propose and negotiate a reward system for meeting or beating the plan. The best approach is to align your interests with those of the business. This might take the form of a three-tier bonus plan: one part based on how well your team performs, another based on user satisfaction, and the other based on how well the company does.

The team bonus helps motivate your team to go the extra mile. User satisfaction is a better measure than post-release defects since it tends to weed out the issues that aren't critical. And the value to the company of a timely delivery of

To select the metrics that encourage the behaviors you need and reward the results you want, you first have to decide *what* you need and want.

a quality product can be measured through increased profitability, reduced costs, or greater market share.

Don't make the incentive all or nothing. This makes it difficult to compromise when conditions are complicated. Don't make it now or never, either—be willing to delay some of the reward until all the results are in. You might distribute the first tier of the bonus when the software is released, the second tier after ninety days of usage, and the third whenever company performance is reported.

## Work the Plan

We all know the best-laid plans can become obsolete from the time you start, so don't just follow the plan; schedule regular reviews. Be quick to note when assumptions aren't met, such as when the software is delivered late, the test environment is not available, or critical defects are blocking progress. Constantly refine and update the plan.

Stay flexible and positive. Don't just huff that the developers were late or the code is bad; instead, point out that your assumptions have not been met and that to stay on track the plan may need modification, such as adding resources, extending the schedule, or reducing coverage or features.

Be just as quick to own your own issues. If you aren't meeting goals because of turnover on your team, over-inflated expectations for automation, or overly optimistic estimates of test execution speed, say so, and propose plans for compensating. Track your metrics continuously, making sure the users and the team know where you stand at all times.

## Train Your Team

While there are always superstars that deserve recognition, it takes an

overall team effort to succeed. Get team members' buy-in, and spread monetary rewards as equally as possible to encourage cooperation. This will create a culture of accountability; team members will regulate each other if they feel someone is dragging everyone down.

Make sure the superstars know their efforts are appreciated within the team and up the management chain. Something as corny as a bulletin board with their pictures on it, a letter of commendation for their files, or a goofy trophy can go a long way.

## Learn and Refine

Finally, don't neglect the post mortem. Review what went right and what could be improved. Identify the metrics that worked, not just in terms of reward but also by creating the right motivations and outcome. Interview your users to see how they thought the project went and what they would like to see changed.

At the end, you will know which metrics motivate—the ones that work. {end}

### Better Software Magazine Publisher's Statement

October 2008

Published in Issue 10-9 November 2008

UNITED STATES POSTAL SERVICE

STATEMENT OF OWNERSHIP, MANAGEMENT, AND CIRCULATION

1. Publication title: Better Software
2. Publication number: 0019-578
3. Filing date: October 2008
4. Issue frequency: January, March, April, May, June, July, September, October, November, December
5. Number of issues published annually: 10
6. Annual subscription price: \$59.00
7. Complete mailing address of known office of publication: Software Quality Engineering, 330 Corporate Way Ste. 300, Orange Park, FL 32073-6214, Clay county
8. Complete mailing address of headquarters or general business office of publisher: Software Quality Engineering, 330 Corporate Way Ste. 300, Orange Park, FL 32073-6214, Clay county
9. Full names and complete address of Publisher, Editor, and Editor in Chief: Publisher: Wayne Middleton, Software Quality Engineering, 330 Corporate Way Ste. 300, Orange Park, FL 32073-6214, Clay county. Editor: Holly Bourquin, Software Quality Engineering, 330 Corporate Way Ste. 300, Orange Park, FL 32073-6214, Clay county. Editor in Chief: Heather Shanholzer, Software Quality Engineering, 330 Corporate Way Ste. 300, Orange Park, FL 32073-6214, Clay county.
10. Owner: C. Wayne Middleton, Software Quality Engineering, 330 Corporate Way Ste. 300, Orange Park, FL 32073-6214, Clay county.
11. Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, or other securities: None.
12. NOT NON PROFIT; DO NOT NEED TO INCLUDE IN STATEMENT
13. Publication title: Better Software
14. Issue date for circulation data below: October 2008
15. Extent and Nature of Circulation:
  - a. Total number of copies (Net press run): 21,695. Actual number of copies of single issue published nearest the filing date: 18,616
  - b. Paid and/or requested circulation (1.) Paid/requested outside-county mail subscriptions stated on form 3541. Average number of copies each issue during preceding 12 months: 17,877. Actual number of copies of single issue published nearest the filing date: 14,430 (2.) Paid in-county subscriptions stated on form 3541. Average number of copies each issue during preceding 12 months: 0. Actual number of copies of single issue published nearest the filing date: 0. (3.) Sales through dealers and carriers, street vendors, counter sales, and other non-USPS paid distribution. Average number of copies each issue during preceding 12 months: 1,119. Actual number of copies of single issue published nearest the filing date: 840 (4.) Other classes mailed through the USPS. Average number of copies each issue during preceding 12 months: 0. Actual number of copies of single issue published nearest the filing date: 0.
  - c. Total paid and/or requested circulation [Sum of 15b. (1), (2), (3), and (4)]: Average number of copies each issue during preceding 12 months: 18,996. Actual number of copies of single issue published nearest the filing date: 15,270.
  - d. Free distribution by mail (Samples, compliment, and other free). (1.) Outside-county as stated on form 3541. Average number of copies each issue during preceding 12 months: 202. Actual number of copies of single issue published nearest the filing date: 199. (2.) In-county as stated on form 3541. Average number of copies each issue during preceding 12 months: 0. Actual number of copies of single issue published nearest the filing date: 0. (3.) Other classes mailed through the USPS. Average number of copies each issue during preceding 12 months: 7. Actual number of copies of single issue published nearest the filing date: 6. (4.) Free or Nominal Rate Distribution Outside the Mail (Carriers or Other Means). Average number of copies each issue during preceding 12 months: 2340. Actual number of copies of single issue published nearest the filing date: 3141.
  - e. Total Free or Nominal Rate Distribution [Sum of 15d. (1), (2), (3) and (4)]: Average number of copies each issue during preceding 12 months: 2,629. Actual number of copies of single issue published nearest the filing date: 3,346.
  - f. Total distribution (Sum of 15c. and 15f.): Average number of copies each issue during preceding 12 months: 21,695. Actual number of copies of single issue published nearest the filing date: 18,616.
  - g. Copies not distributed. Average number of copies each issue during preceding 12 months: 0. Actual number of copies of single issue published nearest the filing date: 0.
  - h. Total (Sum of 15f. and g.): Average number of copies each issue during preceding 12 months: 21,695. Actual number of copies of single issue published nearest the filing date: 18,616.
  - i. Percent paid and/or requested circulation (15c. divided by 15f. times 100). Average number of copies each issue during preceding 12 months: 87%. Actual number of copies of single issue published nearest the filing date: 82%.
16. Publication of Statement of Ownership. Publication is printed in the November 2008 (10-9) issue.

I certify that the statements made by me above are correct and complete.

Wayne Middleton, Publisher

## Index to Advertisers

Aculis	www.aculis.com	16
Agile 2008 Conference	www.Agile2008.org	44
Agitar	www.agitar.com	Inside Back Cover
Better Software Conference 2009	www.sqe.com/BetterSoftwareConf	17
Better Software Magazine	www.BetterSoftware.com	42
Cognizant	www.cognizant.com	12
Hewlett-Packard	www.hp.com/go/software	Back Cover
Mountain Goat Software	www.mountaingoatsoftware.com/better	19
Rally Software	www.rallydev.com	11
Rally Software	www.rallydev.com/bsm	21
Ranorex	www.ranorex.com	32
Seapine	www.seapine.com	1
SQE Agile Training	www.SQETraining.com/Agile	33
STAREAST 2009	www.sqe.com/STAREAST	2
Stevens Institute of Technology	www.stevens.edu/sse	43
StickyMinds.com	www.stickyminds.com	45
TargetProcess	www.targetprocess.com	41
TechExcel	www.techexcel.com	5
VersionOne	www.versionone.com	Inside Front Cover

### Display Advertising

Shae Young syoung@sqe.com

### All Other Inquiries

info@bettersoftware.com

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



# Trying to be **agile** when your Java code is **fragile**?



Feeling the pressure to release software faster?  
Are you bringing new features to market as  
quickly as your business demands?

If enhancing or extending your Java application  
feels risky – if you need to be agile, but instead  
find yourself hanging on by a thread – AgitarOne  
can help.

With AgitarOne's powerful, automated unit  
testing features you can create a safety net of  
tests that detect changes, so you know instantly  
when a new feature breaks something.

Now you can enhance and extend your Java  
applications – fast and without fear.

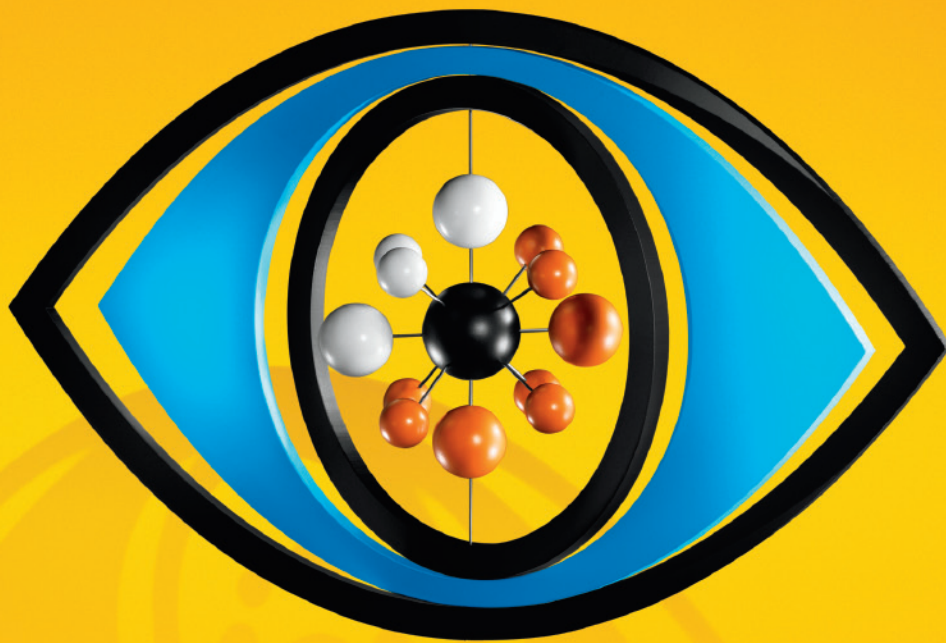
AgitarOne's interactive capabilities for  
exploratory testing makes it easy to test your  
code as you write it.

Whether you're building a new application,  
adding new features to existing software, or  
simply chasing a bug, AgitarOne can help you  
stay a jump ahead.



Visit [www.agitar.com/bettersoftware](http://www.agitar.com/bettersoftware) to request a free evaluation.





ALTERNATIVE THINKING ABOUT QUALITY MANAGEMENT SOFTWARE:

## Make Foresight 20/20.

Alternative thinking is "Pre." Precaution. Preparation. Prevention.  
Predestined to send the competition home quivering.

It's proactively designing a way to ensure higher quality in your  
applications to help you reach your business goals.

It's understanding and locking down requirements ahead of  
time—because "Well, I guess we should've" just doesn't cut it.

It's quality management software designed to remove the  
uncertainties and perils of deployments and upgrades, leaving  
you free to come up with the next big thing.

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

