YOU ~~WANT~~ NEED THE BEST CODE POSSIBLE

HIGH COMPLEXITY IS ... BAD
LOW COVERAGE IS ... BAD
INSECURE CODE IS ... BAD

THERE'S ONLY ONE TOOL FOR HELP IN ALL THESE AREAS.

THAT'S RIGHT. JUST ONE.

McCabe IQ
Enterprise Edition

Advanced Quality Analysis

Stringent Code Coverage

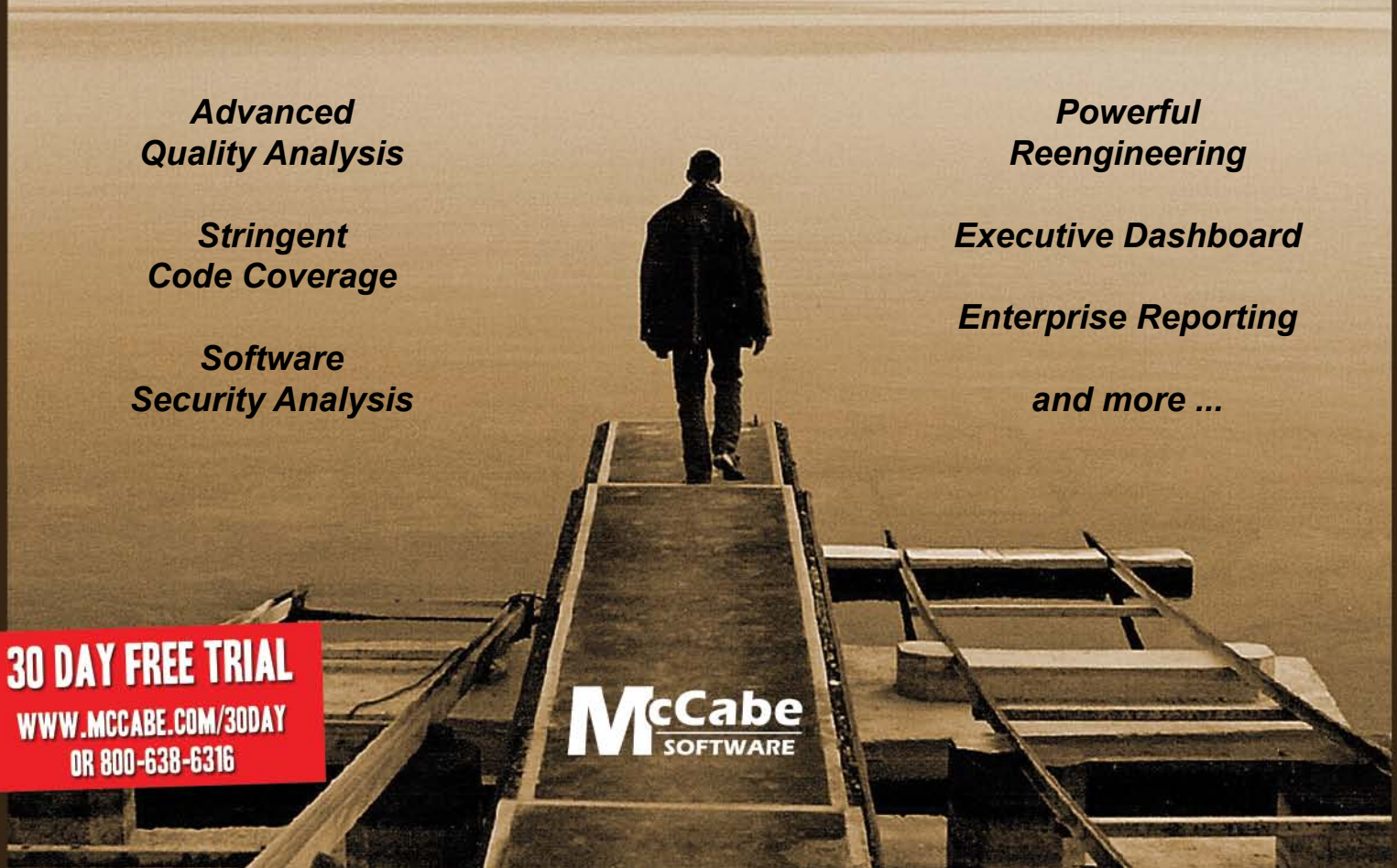
Software Security Analysis

Powerful Reengineering

Executive Dashboard

Enterprise Reporting

and more ...



30 DAY FREE TRIAL
WWW.MCCABE.COM/30DAY
OR 800-638-6316

July/August 2010 $9.95 www.StickyMinds.com

BETTER SOFTWARE

The Print Companion to StickyMinds.com

BETTER BY DESIGN
4 test data design strategies

IT VS. BUSINESS
BAs to the rescue

Slicing Requirements for Agile Success

STAR WEST
SOFTWARE TESTING ANALYSIS & REVIEW
Sept. 26– Oct. 1, 2010
San Diego, CA

TAKE OUR VIRTUAL TEAM SURVEY!

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

Erik Huddleston
CTO
Inovis

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

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

www.rallydev.com | 1.866.348.1552 | info@rallydev.com

© 2010 Rally Software Development Corp

RALLY SOFTWARE

SCALING SOFTWARE AGILITY

www.seapine.com/betterswift
Satisfy your quality obsession.

Save time while protecting software quality.

© 2009 Seapine Software, Inc. All rights reserved.

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

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

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

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

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

Seapine Software™

QA Wizard® Pro Automated Testing

Seapine CM® Change Management

Surround SCM® Configuration Management

TestTrack® Studio Test Planning & Tracking

TestTrack® TCM Test Case Management

TestTrack® Pro Issue Management

TechExcel

ALM Your Way

DevSuite

A Modular Approach to ALM

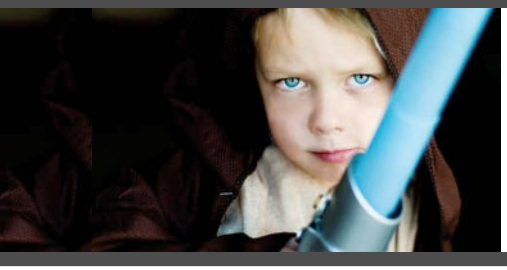
- DevSuite can be purchased and deployed as a complete ALM solution or as individual modules
- Convenient bundles provide solutions for quality management (DevTest Studio) and Agile development (Agile Studio)
- Individual modules and bundles can be easily expanded when you need additional functionality

Agile Ready

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

Download your free 30-day trial of DevSuite now at: www.techexcel.com

Call us at: 1-800-439-7782



Original Software

What Yoda Can Teach Us About Application Quality Management

It's no Jedi mind trick! 5 black holes to avoid for successful application delivery

Executive Summary

A long time ago, in a galaxy not too far away, the very first CHAOS Report published by the Standish Group generated worldwide attention by its claim that 40% of IT projects failed and that these failings were costing the US economy $140 billion each year. Ten years later, matters had improved somewhat with only half as many projects failing, but worryingly 53% were late, over-budget or not meeting their objectives. Now, within a mere 5 years, the number of failed projects is back on the rise; the 2009 Standish Group CHAOS report indicates that nearly 25% projects are doomed.

The quality of application delivery is at the heart of many of the challenges faced in IT projects, and this paper reviews some of the most common pitfalls and pain points that often beset development projects. With the help of Yoda, Obi Wan and others from the Star Wars cast, we will learn how best to avoid these challenges and deliver your projects on time, on budget and most importantly with quality.

The Black Hole No. 1: Walking before you crawl

Obi-Wan: *How long will it take before you can make the jump to light speed?*

Han Solo: *Travelling through hyperspace ain't like dusting crops, boy! Without precise calculations we could fly right through a star, or bounce too close to a supernova and that'd end your trip real quick, wouldn't it?*

It is natural to focus on the eventual goal; the application that will be built and that will deliver the projected business benefits. However, it is equally important to focus on the quality of that deliverable, right from the project's inception. Fail in this and you will face abandoned projects, missed deadlines and an application that may be implemented but will forever after be associated with instability and high maintenance costs.

The first essential step is to recognize this fact and to put application quality and its management at the heart of all your development efforts. If you do not believe this or do not believe you can, then failure is much more likely than success.

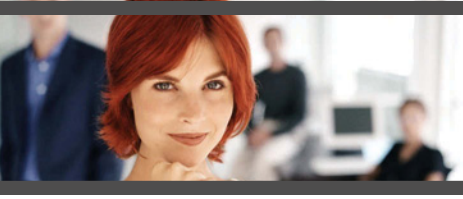
The Black Hole No. 2: QA as a silo

Obi Wan: *The force is what gives a Jedi his power. It is an energy field created by all living things. It surrounds us and penetrates us. It binds the galaxy together.*

The same could be said for quality management. It should be an energy field, created and sustained by all involved in the development process, linking all living parts of the lifecycle, the requirements, code, the build, the test steps, the defects, the regression pack, everything; binding all aspects together and giving us the power of visibility and foresight throughout every stage of the development.

More commonly though, test teams seem to exist in serene isolation: isolated not only from other parts of the development and delivery effort, but also from each other. Frequent status meetings are normal, with the focus on the gathering of historical data rather than forward planning. Similarly, communication with other key teams is often dysfunctional. Defects are reported with a 'fire and forget' mentality. This is fine if you are trying to shoot down an enemy star ship but not so clever when building an IT application, as development is a key partner in application delivery.

To date, test management products have reinforced rather than broken down the potential dangerous isolation of QA teams. They have taken a narrow view of QA with a focus on requirements, tasks and defects when what is needed is a solution that can embrace QA across the project disciplines and integrate into essential infrastructure tools such as change management.



Original Software

The Black Hole No. 3: Lack of visibility and out of date information

The Emperor: *You've paid the price for your lack of Vision: If you will not be turned you will be destroyed.*

Understanding the current project status, the trends in the progress and the implications for resources, target dates and costs are vital to making the correct decisions. The information also needs to be available instantly. If gathering status information across all the project disciplines takes a week, the number of hours potentially burnt in the wrong activities becomes alarming. This could destroy your chances of keeping the project on-track, making your doomed project just another statistic on next year's CHAOS report!

To help you to keep your finger on the pulse of your development lifecycle, you need instant access to key information by the most appropriate and powerful device. Printed reports should be at the bottom of the pile given that they are out of date the moment they are created, PCs and web access are better, and personal devices such as smart phones or Apple iPads at the top of the heap.

The Black Hole No. 4: Unnecessary re-work

General Madine: *Is your strike team assembled?*

It might be the same team that has been used on previous projects, but with many application quality management solutions, users have to be set up over and over again for each project. All too often, although you have your team assembled and ready to go, there is still a frustrating amount of work to do in setting up users, permissions and calendars etc. When evaluating AQM solutions take into account simple time-saving factors and where possible choose an option where users need only be set up the once and can then be assigned to multiple projects.

The Black Hole No. 5: Not supporting all types of working practice

Yoda: *Decide you must, how to serve them best.*

Enterprise Agile versus waterfall and the challenges of heterogeneous environments are increasingly becoming hot talking points and many organizations are working with a variety of platforms and methodologies. So how do you successfully bring together teams that work in different ways and on multiple projects?

With more complexity in IT projects and a need to respond faster to changing markets, development teams have had to adapt the way they work, often utilizing different methodologies on different projects in order to support the dynamic nature of their businesses. If the quality management solution does not support the way that they work, you will encourage maverick teams working outside of the 'Alliance'. Make sure that your AQM solution empowers your teams and allows the flexibility to aid and not impede them.

Conclusion

In this paper, we have looked at ten galactic black holes that projects can get sucked into, turning them off-course. Application Quality Management is not some mysterious Jedi art – most of this is just common sense. You already know what is required in application delivery, you just need to 'use the Force' and remember these black holes when selecting technology to assist you in achieving your destiny.

'Size matters not' says Yoda, but what's important is effective project planning and organization, addressing complexity and empowering different working practices, ensuring good collaboration and communication and maintaining control and visibility throughout. Quality cannot be simply bolted on at the end of a development. It must be embraced from the start and be part of the entire development ethos and infrastructure.

Unfortunately current market-dominating products do not meet these fundamental requirements and only support and reinforce the approach of test management in a silo. Don't become one of the negative statistics in the CHAOS report. In order to deliver a successful solution that meets all the demands of the business, we need to a take a holistic view of quality in the application delivery process.

In the words of Yoda: 'Mind what you have learnt. Save you it can'.

This is a cut down version of a longer Original Insight. To view the full paper and read about ten black holes, visit: http://www.origsoft.com/products/qualify/docs/what_yoda_can_teach_us_about_aqm.pdf

Quotes from Star Wars film characters are accredited to Lucas Films.
All other trademarks are properties of their respective owners,

BETTER SOFTWARE

Volume 12, Issue 4 • July/August 2010

CONTENTS

Take the *Better Software* magazine digital survey. This month's topic Virtual Teams. pg. 15

features

16 COVER STORY

SLICING REQUIREMENTS FOR AGILE SUCCESS

Agile teams need to analyze product requirements in enough detail to build, test, and deliver the right requirements in short time frames. For the many teams that struggle to define "just enough, just in time" requirements, here's help.

by Ellen Gottesdiener and Mary Gorman

22 THE SYSTEM BEHIND THE BEHAVIOR

Most managers are familiar with using feedback and coaching to help people improve their performance at work. But those aren't the only tools managers have. Sometimes, the most effective way to change individual behavior is to change something in the system.

by Esther Derby

26 FOUR PATHS TO PREDICTABLE, REPEATABLE, REUSABLE TEST DATA

Modern applications operate in highly integrated environments, and critical systems rely on massive amounts of data that likely contain sensitive information. Discover useful strategies for preparing your baseline, handling interfaces, designing input data, and planning for output results.

by Linda Hayes

columns

9 TECHNICALLY SPEAKING

BROKEN WINDOWS, BROKEN PROJECTS • *by Lee Copeland*

A social experiment in the '80s found "Vandalism can occur anywhere once communal barriers are lowered by actions that seem to signal that 'no one cares.'" The same can be said for our software projects.

10 INSIDE ANALYSIS

US AGAINST THEM • *by Doug Goldberg*

How well does creating an opposing force serve to deliver on shared objectives within the same organization? A stronger argument may be to teach both business stakeholders and delivery personnel to reach across organizational boundaries to share not only the vision but also the methods used to achieve it.

35 CAREER DEVELOPMENT

SO, YOU WANT TO BE A CONSULTANT? • *by Fiona Charles*

Many practitioners see becoming a consultant as their ultimate career goal. But what does it mean to be "a consultant"? In this email to an aspiring consultant, Fiona Charles (a consultant for more than fifteen years) discusses different consulting approaches and describes how working for a consulting firm can help you get there.

in every issue

Mark Your Calendar **4**

Contributors **6**

Editor's Note **7**

Virtual Resource Shelf **13**

Digital Survey Results **15**

Product Announcements **33**

FAQ **34**

Ad Index **36**

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

Visit www.BetterSoftware.com or call 800.450.7854.

MARK YOUR CALENDAR

SQE TRAINING
www.sqetraining.com

training weeks

www.sqetraining.com/public

Testing

September 13–17, 2010
Washington, DC

October 18–22, 2010
San Francisco, CA

November 8–12, 2010
Tampa, FL

software tester certification

www.sqetraining.com/certification

September 14–16, 2010
Minneapolis, MN
Salt Lake City, UT

September 21–23, 2010
Philadelphia, PA
St. Louis, MO

September 26–28, 2010
San Diego, CA

September 28–30, 2010
Indianapolis, IN

October 5–7, 2010
Baltimore, MD
Toronto, ON

conferences

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

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

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

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

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

BETTER SOFTWARE

Publisher
Wayne Middleton

President
Drew Thoeni

Vice President of Publishing
Holly N. Bourquin

Editor in Chief
Heather Shanholtzer

Editorial

Managing Technical Editor
Lee Copeland

Online Editor
Francesca Matteu

Online Editor
Joseph McAllister

Production Coordinator
Cheryl M. Burke

Design

Creative Director
Catherine J. Clinger

Advertising

Senior Advertising Sales Manager
Shae Young

Sales Manager
Sonia Lavin

Production Coordinator
April Evans

Circulation and Marketing

Circulation Coordinator
Jamie Green-Gago

Marketing Coordinator
Stephanie Fender

A PUBLICATION OF SOFTWARE QUALITY ENGINEERING

SOFTWARE QUALITY ENGINEERING
ASBPE American Society of Business Publication Editors
fma

CONTACT US
Editors: editors@bettersoftware.com
Subscriber Services: info@bettersoftware.com
Phone: 904.278.0524, 888.268.8770
Fax: 904.278.4380
Address:
Better Software magazine
Software Quality Engineering, Inc.
330 Corporate Way, Suite 300
Orange Park, FL 32073

Ranorex®

Automate Your UI Testing with Ranorex

Object-based Capture & Replay Editor

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

Automated Testing of Web & Windows Applications

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

Maintainable UI Object Repositories

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

Write tests in C#, VB.NET and IronPython

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

Get your 30-day Trial
www.ranorex.com

Watch Demo Video
www.ranorex.com/demo

Contributors

Fiona Charles is a Toronto-based test consultant and manager with thirty years of experience in software development and integration projects. Fiona is the editor of *The Gift of Time*, featuring essays by consultants and managers of various professions about what they've learned from Gerald M. Weinberg. Through her company, Quality Intelligence, Inc., Fiona works with clients in diverse industries to design and implement pragmatic test and test management practices that match their unique business challenges. Her experiential workshops facilitate tester learning by doing, either on the job or at conferences. Contact Fiona via her Web site at www.quality-intelligence.com.

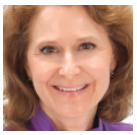
Lee Copeland has more than thirty years of experience in the field of software development and testing. He has worked as a programmer, development director, process improvement leader, and consultant. Based on his experience, Lee has developed and taught a number of training courses focusing on software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

Esther Derby is a regular StickyMinds.com and *Better Software* magazine contributor. Recognized as one of the world's leaders in retrospective facilitation, she often receives requests to work with struggling teams. Esther is one of the founders of the AYE Conference and has presented at STAR*EAST* and the Better Software Conference. Esther is the co-author of *Agile Retrospectives: Making Good Teams Great*. You can read more of her articles on the wonderful world of software at estherderby.com and on her blog at estherderby.com/weblog/blogger.html. Her email is derby@estherderby.com.

A senior business analyst in Dallas, Texas, Doug Goldberg has fifteen years of experience as an analyst in application development for financial, healthcare, and technology companies. He has also programmed Java/J2EE. Doug is an avid business analyst mentor both online and in person, a facilitator for CBAP study groups, an active guest blogger, and the current VP of professional development for the Dallas Chapter of the IIBA. You can reach Doug at douggoldberg@gmail.com.

Mary Gorman, CBAPTM, is senior associate at EBG Consulting and helps project teams explore, analyze, and build robust business and system requirements models. Mary serves on the Business Analysis Body of Knowledge committee of the International Institute of Business Analysis and is the leader of the elicitation subcommittee. She can be reached at mary@ebgconsulting.com and ebgconsulting.com. Mary and co-author Ellen Gottesdiener thank Susan Block, Mike Cohn, Chris Matts, Tom Poppendieck, Jeff Sutherland, and Bill Wake for their review and incisive comments on a draft of their article "Slicing Requirements for Agile Success."

EBG Consulting principal consultant and founder Ellen Gottesdiener helps business and technical teams collaborate to define and deliver products customers value and need. Author of two acclaimed books, Ellen works with global clients and speaks at industry conferences. Learn more from her articles, tweets, blog, and free eNewsletter and find resources on EBG's website, ebgconsulting.com. Contact Ellen at ellen@ebgconsulting.com.



A frequent industry speaker and award-winning author on software quality, Linda G. Hayes is a founder of Worksoft, Inc., developer of next-generation test automation solutions. 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, author of the *The Automated Testing Handbook*, and co-editor (with Alka Jarvis) of *Dare to be Excellent* on best practices in the software industry. You can contact Linda at linda@worksoft.com.

Dawn Haynes has more than twenty years of experience supporting, administering, developing, and testing software and hardware systems. She is a senior trainer and consultant for PerfTestPlus and a director of the Association for Software Testing. A highly regarded trainer of software testers, Dawn blends experience and humor to provide testers of all levels with tools and techniques to help them generate new approaches to solving common and complex software testing problems. Dawn is particularly passionate about improving the state of performance testing across the industry.

BETTER
SOFTWARE
MAGAZINE
Summer
Promotion!
$14.95 Print
Subscription*
www.bettersoftware.com/SUM2010
*U.S. Only

BETTER
SOFTWARE
MAGAZINE
Summer
Promotion!
$14.95 Print
Subscription*
www.bettersoftware.com/SUM2010
*U.S. Only

Taking Projects Down to *The Wire*

My husband and I just spent a few weeks watching DVDs of *The Wire*—in my opinion the best TV series ever made. Set in the inner city of Baltimore, Maryland, *The Wire* explores how members of a society interact and influence one another. Each season, the story is told from a different perspective—drug dealers and users, the port system, city government, the education system, and the city newspaper—just a few of the institutions that intertwine to make up the fabric of a society.

One of the aspects of the show that I found the most fascinating was the human interaction across all facets of society, from the gang bangers to the teachers to the city council members, and how the actions of one group inevitably affects the actions of the others.

The Wire and its complicated interrelationships sprang to mind as I read this issue's Technically Speaking, "Broken Windows, Broken Projects." In this column, Lee Copeland describes the theory that if a window is broken and not repaired, more vandalism is sure to follow as the broken windows are an indication that no one cares about the property's upkeep.

Lee extends this theory to the "broken windows" on our projects—the missed deadlines, rampant defects, etc.—that, if ignored, signal that no one cares about the success of the project. Fortunately, the solution to fixing broken windows on projects is usually much less complex than repairing all that ails a community the size of Baltimore.

Another article in this issue that delves into how behavior influences project outcomes is Esther Derby's "The System Behind the Behavior." We've all worked on projects with a "bad apple," that one person who always seems to throw a wrench in the works of an otherwise functioning team. However, as Esther explains in her article, sometimes the problem doesn't lie with the individual but within the system itself—a problem with the way the whole group is working.

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

Happy Reading!

Heather Shanholtzer
HShanholtzer@sqe.com

Microsoft® Visual Studio®

NOW I CAN PUT "NO-REPRO" TO BED.

AUTOMATICALLY ATTACH CONTEXT TO BUGS.

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

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

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

WHAT WILL YOU DO WITH VISUAL STUDIO 2010?

Visit **www.almcatalyst.com/test**

Download Test Professional and Lab Management.

Not sure how Visual Studio 2010 can help your team?

Connect with an ALM expert
Sign up for an ALM workshop

Microsoft

Broken Windows, Broken Projects

Just as broken windows in neighborhoods can lead to vandalism, "broken windows" on projects can lead to failure. Trust is key.

by Lee Copeland | lcopeland@sqe.com

In March 1982, *The Atlantic* magazine ran an article titled "Broken Windows" by George L. Kelling and James Q. Wilson. [1] The authors of this now famous article wrote, "Social psychologists and police officers agree that if a window in a building is broken and is left unrepaired, all the rest of the windows will soon be broken." One broken window, left unrepaired, is a signal that the building is abandoned and that no one cares, so breaking more windows means nothing. The authors continue, "Vandalism can occur anywhere once communal barriers—the sense of mutual regard and the obligations of civility—are lowered by actions that seem to signal that 'no one cares.'"

To test this theory, Philip Zimbardo, a Stanford psychology professor, had an automobile without license plates parked with its hood raised on a street in the Bronx. Within ten minutes of its "abandonment," the car was attacked and stripped. First, the battery and radiator were removed, and within twenty-four hours almost everything of value had been taken. Random destruction then began with the smashing of windows and the ripping of upholstery. Within a few more hours, the car had been turned upside down and totally destroyed.

To test the claim that this behavior could "occur anywhere," Zimbardo performed the same experiment in sedate Palo Alto, California. This car fared somewhat better, remaining untouched for more than a week. Then, Zimbardo hit the car with a sledgehammer and within a few hours, this car also was looted, turned upside down, and destroyed.

To put the broken windows theory to the test, Kelling was hired as a consultant by the New York City Transit Authority. The subway system was cleaned—specifically targeting graffiti removal. In 1990, William Bratton, an admirer of Kelling, took over the Transit Police and implemented a zero-tolerance for fare dodging and easier processing of those arrested. These strategies, among others, became part of Mayor Rudy Giuliani's "quality of life" initiative. [2] Was it effective? A later study of crime trends in New York City showed the rates of both petty and serious crime fell suddenly and significantly after these actions were instituted.

Most of us have experienced our fair share of project broken windows: impossible deadlines, shipping a product filled with disaster-in-waiting defects just to meet an arbitrary schedule, firing the tester for finding too many defects, bogus incentive schemes, declaring a project "done" when it's not and phony-ing up documents to prove it, and the ever-popular adding more people to a late project.

Windows get broken on every project. However, it's vital not to let them remain broken for long, as the price can be high if they are ignored. To paraphrase Kelling and Wilson's statement: A violation of trust, left for a time, is a signal that the project has been abandoned, that no one cares, and so violating additional trust means nothing. Once the sense of mutual regard, obligations of civility, and trust are lowered by actions that signal that no one cares, why should anyone continue to care? All it takes is one broken window—the project manager imposes an unrealistic schedule; analysts skip the details of requirements in an extremely complex part of the system; designers and developers ignore future system needs for different parameters, data, algorithms, languages, or hardware devices; testers just focus on creating large numbers of tests rather than creating effective tests; users pay cursory attention to acceptance testing. When it becomes clear that no one cares then, quickly, no one cares.

"Windows get broken on every project. However, it's vital not to let them remain broken for long as the price can be high if they are ignored."

A key tool for repairing broken windows is trust. Trust comes from integrity—a steadfast adherence to a strict moral or ethical code, ongoing communication of vision and values synchronizing the team's direction, valuing team members as equal partners in project success, focusing on shared goals rather than personal agendas, competency with the technical processes, and doing what's right, regardless of personal risk.

Surely, Brad Appleton was right when in 2004 he wrote, "The most important and useful thing that the initial iteration needs to build is ... trust!" [3] {end}

Sticky Notes

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

- References

Us Against Them

The IT and business stakeholder relationship can be contentious. Business analysts are positioned to foster collaboration between the two.

by Doug Goldberg | douggoldberg@gmail.com

In some environments, going up against an adversary is not only a good thing to do—it's expected in order to win. However, in the sometimes contentious world of development, it's almost guaranteed to cause a project to fail.

There is a long history of business organizations feeling shortchanged by IT organizations based on perceived value and incorrect assumptions, such as when there is no comprehension of the effort required to implement what is thought to be a simple change. These are dangerous characterizations, and without good communication between those who need support from a technical organization and those who provide it, there is a great chance that success will not be achieved.

Fortunately, the role of the business analyst is becoming more prevalent and better understood. While most people perceive a business analyst as necessary to gather and document requirements, many overlook an analyst's ability to function as a liaison between business stakeholders and IT delivery personnel, which creates a positive relationship between the two. This is a critical capability and one that requires advanced communication skills that include active listening, relationship building, and collaborative facilitation.

A business analyst is in a unique position both to understand the needs of stakeholders and to protect the interests of the delivery organization—and it really doesn't matter to which side of the business/IT "fence" the analyst reports. The analyst's skills provide a means to bridge gaps by exposing differences with explanation and facilitation, fostering constructive dialogue and collaboration, and breaking down barriers to success. There is probably no other position in an organization that understands the strategic goals of business operations while concurrently recognizing the pressures and limitations of IT delivery. That means that, in the end, the organization's executives have their specific interests protected at the individual project level, which, in turn, ensures a healthy balance in the handling of development lifecycle pressures.

"While most people perceive a business analyst as necessary to gather and document requirements, many overlook an analyst's ability to function as a liaison between business stakeholders and IT delivery personnel, which creates a positive relationship between the two."

Fostering a collaborative approach to bring stakeholders' often conflicting interests to bear, the analyst can undertake exercises and utilize techniques to pull these forces together to achieve a common goal. It is the common goal that is often forgotten within an organization when business and development teams head into the project arena for a battle that was completely unnecessary in the first place.

There are a couple of specific approaches that an analyst can take to bring harmony to an organization.

IT-side Analyst

There are several things that can help strengthen the relationship with business-side partners. For instance, providing an explanation of seemingly superficial problems brings a more thorough understanding to the problem domain. Often, what appears to be a simple fix is in fact much more complex. This is a divisive point between business and IT, as one side feels it isn't getting value for dollars spent and the other claims that there is no understanding of what is really involved. Without some cursory handholding and explanation, the division broadens and tensions build.

This scenario is less evident in an agile environment that has the business stakeholder embedded into the development team. In IT shops that employ the waterfall methodology, it can be a huge problem. Often, there is a perception of someone just hammering out a little code and "presto!" the change appears ready to use in production. Without realizing that there might be additional analysis, design sessions, test creation and execution, required documentation, and release or deployment activities that all require time from resources, the business stakeholder may have trouble understanding why IT takes so long to produce changes.

The lack of collaboration and communication exacerbates the perceptions and emotions that grow out of the lack of knowledge. Reaching across boundaries to collaborate helps to build relationships that foster trust. This is not an over-

night approach that yields immediate results, but more of one that is created iteratively. Moreover, it can require explicit effort by an analyst to enlighten peers and stakeholders on the business side, as well as explicit statements that one is taking an extra step to ensure understanding. Trite as this direct communication might sound, it is a huge stepping stone to enhancing relationships and building the trust that binds the IT and business partners together.

Finally, an IT analyst should always approach every communication with a "What can I do for you?" customer service attitude, and the customer must know, based on the IT analyst's actions, that his concerns are encompassed and protected by the IT analyst while being balanced against IT capabilities.

Business-side Analyst

In a business-side role, much of the same must occur but in reverse. The business analyst not only functions to capture and understand operational and strategic initiatives but also is the liaison who communicates those initiatives to the IT organization with knowledge about what that group must do to accomplish the tasks at hand. Again, this is often where things break down due to lack of knowledge.

The business-side analyst has an obligation to reach out and ask about what occurs following business requirements turnover in IT, and this very simple action eradicates a lot of built-up tension on the IT side. It shows that someone cares enough to find out what IT is all about and its capabilities within the constraints that are often invisible to business partners. While it's not the business partner's job to know all this, it is a liaison's obligation in an environment in which all teams work together to achieve goals.

It is also important from the business perspective to have a clear understanding of IT's expectations on requirements. Being able to factor this knowledge into the business requirements deliverables removes a tremendous amount of rework due to inadequate requirements and makes everyone's life less stressful.

As a final rule, the business-side analyst must recognize the IT organization as the primary customer and recipient of the business product (the requirements deliverable). An attitude of excellent customer service builds the relationship in the opposite direction. Actively seeking feedback on deliverables and interactions helps to solidify the team cohesion necessary for more productive output.

If you are a business analyst or you work with one in either of the above situations, try exploring a few of these techniques to begin to foster better communication—even when the job doesn't require it. After all, each of us is trying to do the right thing and, while that looks different to different people, it's usually part of a common goal. Breaking down the barriers that keep us working separately can lead to some surprisingly positive results and help create a very productive and rewarding workplace environment. **{end}**

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

WANT TO RECEIVE COMPLIMENTARY COPIES OF SOME OF THE LATEST BOOKS ON SOFTWARE DEVELOPMENT?

Join the StickyMinds.com Book Review Program! If you're an experienced software professional who likes to read and thrives on sharing opinions, join our unique book review program that caters exclusively to the software development community!

For an application or more information, contact Cheryl M. Burke: cburke@sqe.com.

YOU ~~WANT~~ NEED THE BEST CODE POSSIBLE

HIGH COMPLEXITY IS ... BAD
LOW COVERAGE IS ... BAD
INSECURE CODE IS ... BAD

THERE'S ONLY ONE TOOL FOR HELP IN ALL THESE AREAS.

THAT'S RIGHT. JUST ONE.

McCabe IQ
Enterprise Edition

Advanced Quality Analysis
Stringent Code Coverage
Software Security Analysis
Powerful Reengineering
Executive Dashboard
Enterprise Reporting
and more ...

30 DAY FREE TRIAL
WWW.MCCABE.COM/30DAY
OR 800-638-6316

McCabe SOFTWARE

Testing Under Pressure?

We Can Help.

Check Out New 2010 Training Courses Now Available!

COMBINE TRAINING WEEK AND SAVE

The more training you take the greater the savings!

Maximize the impact of your training by combining courses in the same location. Combine a full week of training for the largest discount! For a complete list of courses available, visit **www.sqetraining.com** or call 888.268.8770 or 904.278.0524 for pairing discount options.

SQE TRAINING
www.sqetraining.com

Choose from 16 specialized training courses:

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
Software Tester Certification—Foundation Level			Mastering Test Design	
Systematic Software Testing*			Performance, Load, and Stress Testing Workshop	
Finding Ambiguities in Requirements	Writing Testable Requirements		Requirements-Based Testing	
Testing with Use Cases**	Risk-Driven Software Testing**		Test Estimation & Measurement	Mastering Test Automation
Testing Under Pressure	Agile Testing Practices		Test Process Improvement*	Free Seminar: 9 a.m. – 11 a.m.
Leadership for Test Managers	Test Process Management		*Note: Free Seminar will be on Thursday in Washington, DC and San Francisco, CA and on Friday in Tampa, FL.*	

*Available in Tampa, FL ** Available in Washington, DC & San Francisco, CA

TESTING TRAINING WEEKS

Sept. 13–17, 2010 *Washington, DC*
Oct. 18–22, 2010 *San Francisco, CA*
Nov. 8–12, 2010 *Tampa, FL*

PMI Project Management Institute R.E.P.

The following courses are PMI® Certified:
- Software Tester Certification—Foundation Level
- Mastering Test Design

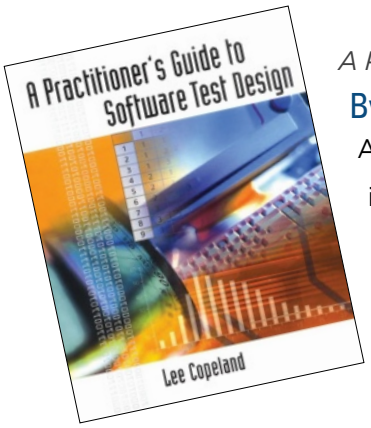
ON-SITE ADVANTAGE

Let SQE Training come to you. For more information about on-site training courses, contact SQE Training at 888.268.8770 or 904.278.0524 ext. 212 or 233 or email ***onsitetraining@sqe.com.***

Virtual **Resource** Shelf

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

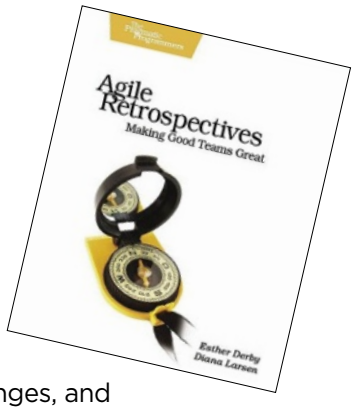
Q: Where can I read more from this issue's authors?



A Practitioner's Guide to Software Test Design

By Lee Copeland

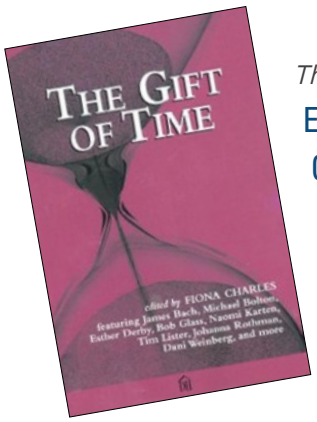
A comprehensive, up-to-date, and practical introduction to software test design.



Agile Retrospectives: Making Good Teams Great

By Esther Derby and Diana Larsen

Learn how to architect retrospectives: designing them specifically for your team and organization, running them effectively, making the needed changes, and scaling these techniques up.



The Gift of Time

Edited by Fiona Charles

Consultants and managers from diverse fields present perspectives on lessons learned from Gerald M. Weinberg.

The Automated Testing Handbook

By Linda Hayes

An introduction to test automation and an overview of key technical concepts.

"Business Rules and Data Requirements"

By Mary Gorman

www.stickyminds.com/s.asp?F=S14265_COL_2

Visit StickyMinds.com to read more of Mary's articles.

DougGtheBA

By Doug Goldberg

www.bridging-the-gap.com/author/douggtheba/

Doug is an active blogger on bridging-the-gap.com, a site devoted to "helping business analysts become leaders and advance their careers."

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

Attend Live, Instructor-led Classes Via Your Computer

New Live Virtual Courses Now Available!

LIVE VIRTUAL TRAINING

NEW Live Virtual Courses:

- » Testing Under Pressure
- » Performance, Load, and Stress Testing
- » Transitioning to Agile
- » Key Test Design Techniques
- » Practical ERP Testing Strategies
- » Essential Test Management and Planning
- » Finding Ambiguities in Requirements
- » Getting Requirements Right the First Time

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

Live Virtual Package Includes:

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

SQE TRAINING

www.SQETraining.com

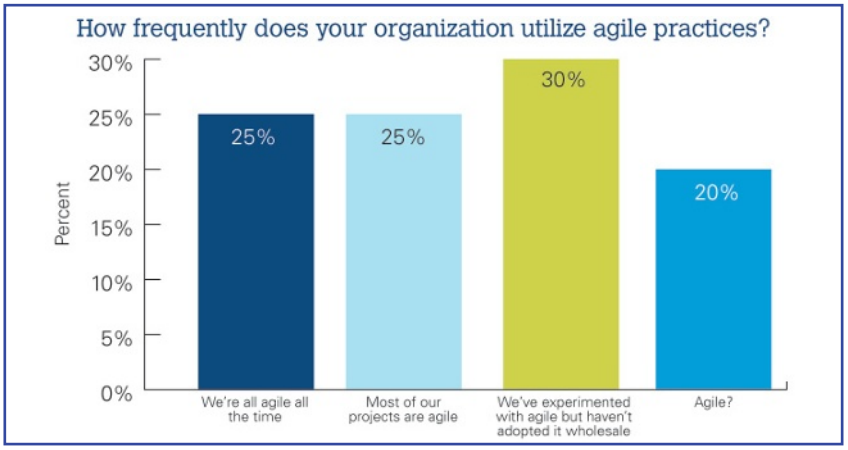
Participate in our new survey:

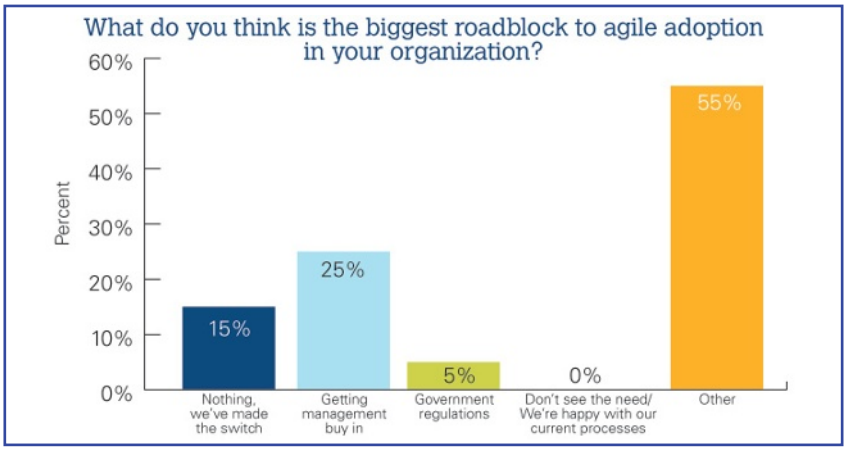
Do You Work on a Virtual Team?

www.stickyminds.com/virtualteam

Last Issue's Survey Results:

Q: Has your organization gone agile?

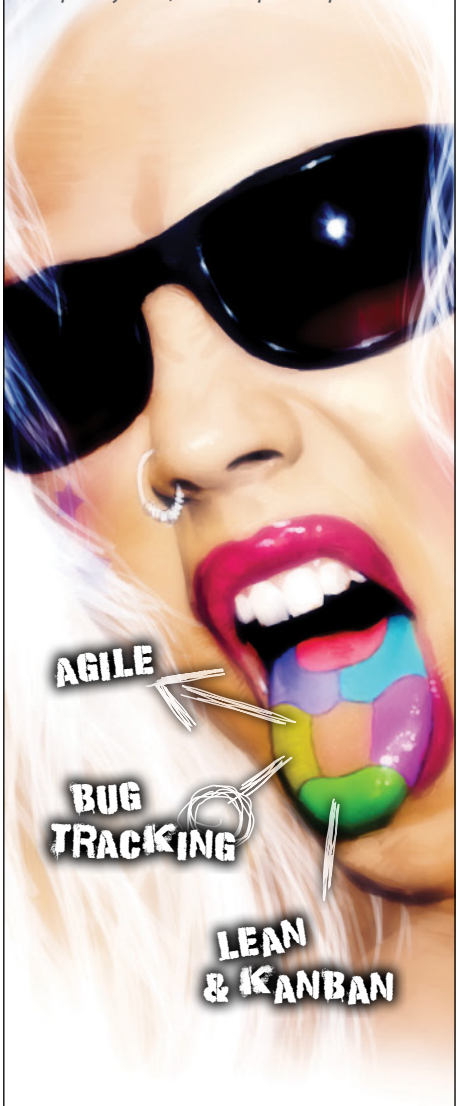




"We are excited...

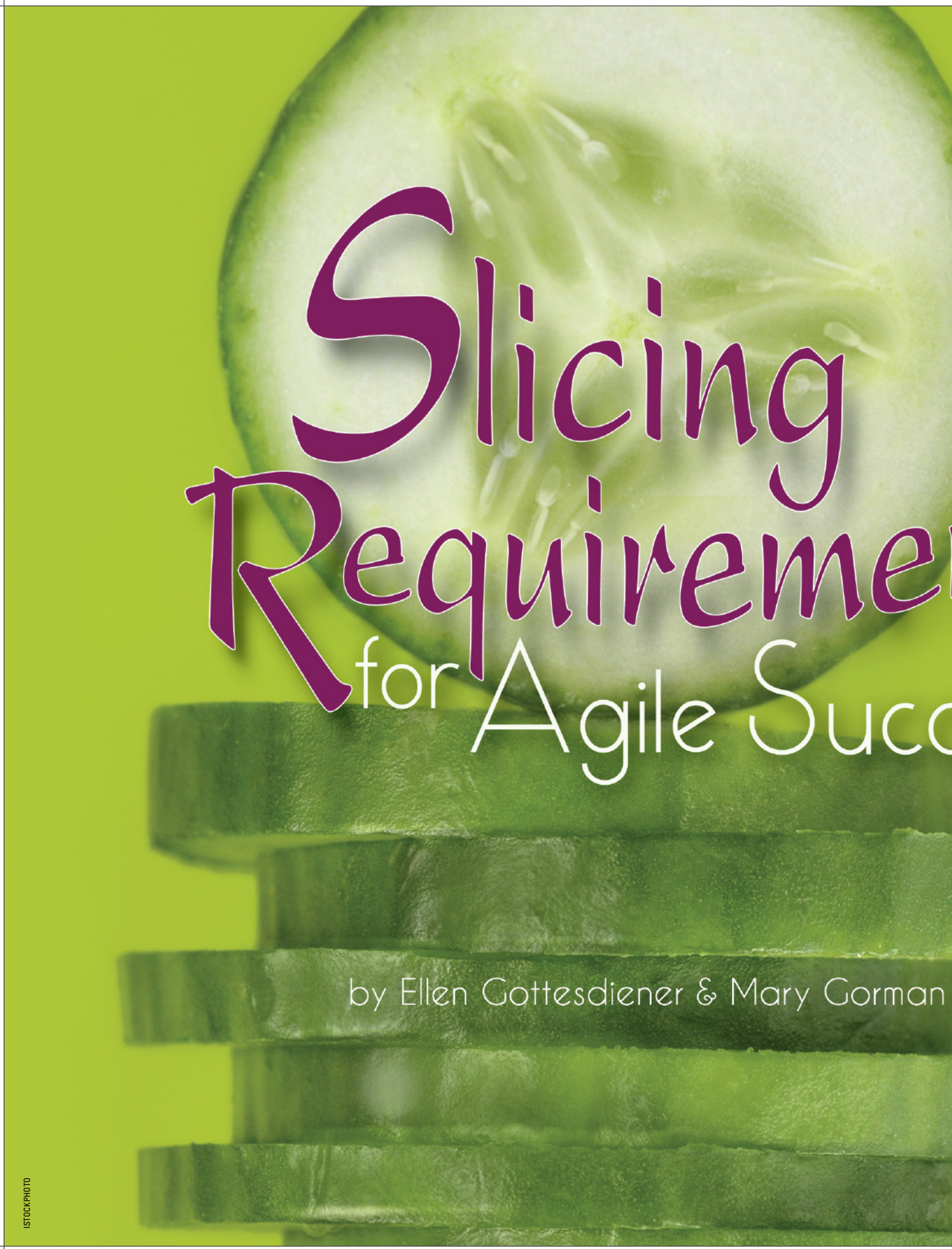
...that Hansoft can handle both schedule driven projects and agile methods like Scrum. The possibility of combining these methods in the same project has really been useful. Hansoft makes our projects more efficient and helps us collaborate much better with partners, customers and suppliers."

General Manager,
Space Systems, Swedish Space Corporation.



HANSOFT

Download a free 2-user solution trial at www.hansoft.se



Slicing Requirements for Agile Success

by Ellen Gottesdiener & Mary Gorman

ISTOCKPHOTO

If you've struggled to find collaborative ways for your team and customer to understand evolving product needs while delivering value on a steady basis, you're not alone. Challenges include accurately estimating the work, readily defining conditions of satisfaction ("doneness" criteria) for requirements, churning on requirements as you begin development, being unable to deliver as promised because new requirements pop up in the middle of development, inability to easily pull requirements into current development work when there is excess capacity, or being unable to meet your delivery commitments.

A typical culprit is "chunky" requirements. They're too big, they aren't clearly prioritized, or the team doesn't understand them in sufficient detail to implement them efficiently.

We've used a variety of "splitting" techniques to ratchet up agile teams' awareness of the importance of working with requirements in the form of user stories. [1] Popular approaches include splitting by business process, database operations (e.g., create, read, update, delete), data groups, low fidelity versus high fidelity, and so on. [2, 3, 4]

Some teams find it hard to apply story-splitting strategies consistently. Others focus on the technical aspects of these strategies rather than base their work on a firm understanding of user needs. And still others neglect nonfunctional requirements, focusing only on functional requirements.

If this sounds like your agile team, stay tuned. We want to share a better way.

Slicing User Story Options Based on Value

Our requirements-slicing technique starts from the assumption that the team and the customer have a shared understanding of the product vision and goals, and they agree on what constitutes value and how value will be measured. Our focus here is on a practical way to pull the requirements out of those high-level needs so that the team and customer can deliver working software in smaller chunks over time.

To do that, teams and business customers need to jointly explore requirements options, identify the most important ones, and slice chunky stories into manageable pieces. The optimal slicing technique would be reusable in all problem domains, leverage the discipline of requirements analysis, be quick to learn and efficient to use, engage product owners (a.k.a. customers), put the "user" back into user stories, directly feed into acceptance tests, and deepen the entire team's knowledge of the requirements.

Based on our experience with the power of small, testable user stories and inspired by the work of Chris Matts and Olav Maasen on real options and feature injection [5, 6], Bill Wake and others on story splitting [1, 2, 3, 4], Jeff Sutherland and others on ready requirements [7, 8, 9, 10], Dean Leffingwell on a lean backlog [11], and Mike Cohn on minimizing team handoffs [12], we've defined a straightforward way to slice user stories.

What Is a Ready User Story?

Ready stories are small, well-understood, valuable user stories. Other names for user stories in this state include right-sized, sliced, iteration-level, queued for development, and analyzed stories.

A ready user story is highly valued by the product owner, needed for delivery in the next release or iteration, and understood well enough by the delivery team to estimate, plan, commit to, and deliver.

When Do You Make User Stories Ready?

You want to get user stories ready just before planning as part of *work-ahead analysis* or *backlog grooming*. You groom your backlog continuously so that items are ready before estimating and planning—whether you use timeboxed iterations or a continuous-flow mechanism such as kanban. The team needs to be ready, with enough requirements knowledge to specify acceptance criteria, reliably estimate the work, and meet delivery commitments. By conducting regularly scheduled, informal agile requirements workshops, the team collaborates with the product owner to groom the backlog for changing requirements and prepare user stories for upcoming delivery cycles.

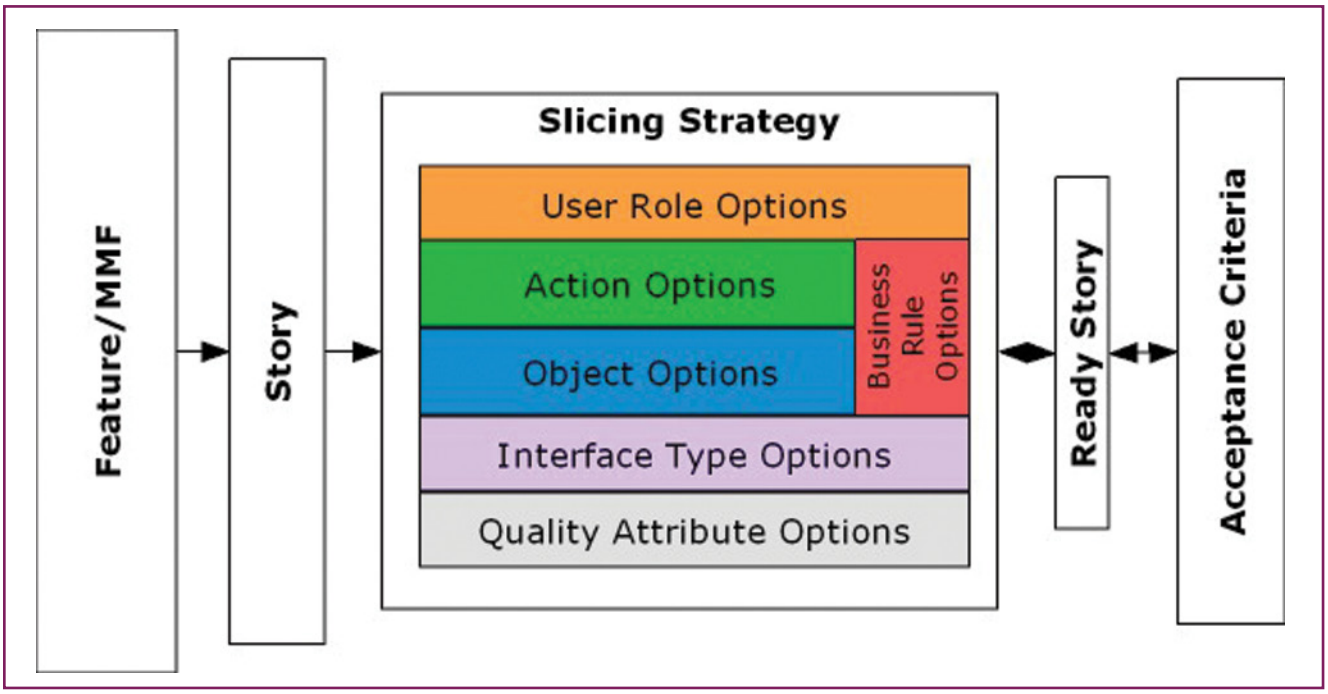


Figure 1: The six slicing elements of a user story

Analyzing User Story Options: Expand-Then-Contract

To get your user stories ready, our story-slicing technique follows an overall pattern of *expand-then-contract*. For each user story, in the expand phase, the product owner first elaborates possible options. Then, in the contract phase, the team and the product owner explore the options and quickly narrow them based on their value; this takes a matter of minutes. The result is a small, concise, thinly sliced story ready for development.

Let's explore this pattern in more detail.

In the expand phase, you begin with a single chunky story and explore the analysis options. Your analysis looks at six elements of the user story:

- Who acts in the *user role*? What are the types and states of that role?
- What *actions* are needed to meet the user's goal?
- What *data objects* are acted on? What are the types and states of those data objects?
- What *business rules* must be enforced for the actions and data objects?
- What *interfaces* are needed? What are their types—manual solutions, physical hardware devices, system-to-system interfaces, or user interfaces?
- What *quality attributes* constrain and control the sliced user story?

In the contract phase, the product owner collaborates with the team to select options using transparent, rational criteria—for example, return on investment, market drivers or events, or risk reduction. She uses prioritization techniques such as a tailored ranking scheme, stakeholder value, voice of the customer, and so on.

Throughout this expand-then-contract process, the team and product owner together gain crucial knowledge. As the product owner assigns a value to each option, she asks the team to gauge the effort and risks for implementing it, heightening her understanding of technical concerns and the development process. As team members discuss the options, they learn about the requirements. Team members question, challenge, and clarify the product owner's filtering criteria, deepening their understanding of the business domain and the requirements that will provide value.

Note that you will find this technique useful for product requirements but not team or technical work, which some teams represent as "technical" stories. And remember: As you proceed through each expand phase, if you discover that any of the six user story elements has no options, simply move to the next element.

Slicing Stories by Analyzing Options: An Example

Let's iterate through the technique again, this time looking at the six elements via a sample story.

You begin by selecting a high-value user story from your backlog. The story might originate from, say, a high-level feature or a minimal marketable feature (MMF). [13] Or, it might have been decomposed from chunkier requirements—what some teams call "epics."

On agile teams, these stories are usually written in the following form:

As a [*user role*]
I need to [*some behavior or action*]
so that [*business value*]

"Your goal is to attain a set of high-value, ready stories that are about equal in size and are sufficiently analyzed so that you can size, plan, and deliver them as promised."

Next, determine if the story is a candidate for slicing.

What makes a user story "too big"? That differs for each team. Your team's size and iteration or release length are factors. Some teams strive to size their stories so that they can complete a single story in a few hours or a day. Other teams have a longer time frame—two to three days at most.

Your goal is to attain a set of high-value, ready stories that are about equal in size and are sufficiently analyzed so that you can size, plan, and deliver them as promised.

Books and Beyond

Figure 1 shows the six elements of a user story.

In our sample, we're building an application for a business that sells products such as books, movies, music, and greeting cards. You select the big, chunky story: "As a *customer*, I want to *buy a product* so that I *can enjoy using it*." Follow along as we explore options and assign a value to each one.

Step 1: User Role Options: Types and States

To analyze the user role, we explore options for the *types* and *states* of the customer. The user role initiates the story and has the goal of obtaining value. The name of the role is based on the user's intention or goal. In the user story, the customer intends to buy; hence, we name the user role "buyer."

Working with the product owner, we define the types of buyers:

User Role Type Options
Individual Buyer
Corporate Buyer
Club Member Buyer
Employee Buyer

These types, along with all the domain terms, should be defined in the product glossary to enable a shared understanding of the domain (also providing a stable basis for domain-specific modeling).

Select for Value

Now ask the product owner to prioritize the user-role type options. Which one has the highest business value for the next iteration? Our product owner wants to focus on income and not slow down delivery with complex buyer types. She selects Individual Buyer.

This is our first use of the expand-then-contract pattern—but definitely not our last.

For the chosen user role type, next explore its possible states or the lifecycle stages it undergoes.

User Role State Options (for Individual Buyer)
New
Existing
Anonymous (can buy without providing a name)
Archived

Select for Value

Now that we've expanded the states, it's time to contract. Again, our product owner needs to narrow her options to the buyer state that will yield the highest immediate value. She selects Individual, Anonymous Buyer.

Step 2: Action Options

To identify all the possible actions that satisfy the selected user role's desired goal, start with the verb in the user story text: "I want to *buy* a product. . ."

Ask the product owner what typically happens (or, if the action is new, what will potentially happen) and what decisions must be made. For the individual, anonymous buyer, the buy action options are as follows:

Buy Action Options
Verify product cost
Calculate tax amount
Calculate total purchase amount
Apply discount
Apply wrapping fee
Arrange for shipping
Secure payment
Adjust inventory
Generate receipt
Post payment to accounts receivable

We suggest writing each action option on a sticky note so that the product owner can move them around. The order of the options is not critical at this point.

Select for Value

The product owner surveys the action options and determines the minimum options needed for the next delivery

cycle. Encourage her to defer actions that are less valuable, occur infrequently, require data not yet built into databases, or are complex and better left until later.

Be sure to analyze any interdependent options—those that might need to be selected together. At the same time, look for independent actions. For example, ask your product owner, "Is it necessary to offer discounts in the next iteration or release, or can that wait?"

To guide the selection, consider the context of the action. For example, is the buyer shopping at a brick-and-mortar store or online? We learn that the immediate need is to support in-store purchases. Our product owner selects four options (checked):

Buy Action Options

- ✓ **Verify product cost**
- **Calculate tax amount**
- ✓ **Calculate total purchase amount**
- **Apply discount**
- **Determine wrapping fee**
- **Arrange for shipping**
- ✓ **Secure payment**
- **Adjust inventory**
- ✓ **Generate receipt**
- **Post payment to accounts receivable**

Step 3: Data Object Options: Types and States

Next, we focus on data objects. Your product owner should identify object options based on her prior selections of user role and actions. In our example, the objects are Product, Payment, and Receipt.

For each object option, identify types or variations. For example, the Product object's types include Book, CD, DVD, and so on.

Select for Value

The team has analyzed the objects and expanded its understanding of the objects' types. The product owner assesses the business value of the options and determines valid combinations. The product owner chooses the highest-value object types (checked items in table 1):

Product Type Options	*Payment Type Options*	*Receipt Type Options*
✓ Book	✓ Cash	✓ Cash receipt
CD	Credit card	Credit card receipt
DVD	PayPal	
Gift card	Purchase order	
Greeting card		
Electronic book reader		

Table 1

There's more. As you did with the user role, you need to explore possible lifecycle states for the objects specific to our story. In our example, the Book object has two states: new or used.

Select for Value

As before, the product owner now contracts the options. Her selection is New Book.

Book State Options

- ✓ **New**
- **Used**

What We've Sliced So Far

We started with the user story: "As a *customer*, I want to *buy a product* ..." We used the expand-then-contract pattern to slice three elements: user role (types and states), actions, and data objects (types and states). The story has been sliced:

"As an *individual anonymous buyer*, I want to *buy a new book ... paying with cash and receiving a cash receipt*."

The story includes four action options: verify product price, calculate total purchase amount, secure payment, and generate receipt.

At this point, you have partially sliced your story. Your team might choose to defer analyzing the three additional slicing options (business rules, interfaces, and quality attributes) until development. But, in our experience, the additional minutes spent exploring these other options before you commit to delivering a story can significantly help in creating a ready story while expanding the team's knowledge of the product requirements. Let's take a look.

Step 4: Business Rule Options

Business rules define constraints and conditions that must be satisfied. As part of developing your story's sliced, high-value actions and objects, you need to elicit business rules that must be enforced. Limiting business rule options to only the prioritized user role, actions, and objects streamlines your analysis work and ensures eliciting "fresh" rules.

Select for Value

The product owner again weighs the business value of these options for the next iteration. She chooses the following (checked):

Business Rule Options

- ✓ **Payment currency must be specific to purchase location**
- **Cash payment denomination amount must not be greater than …**
- ✓ **Payment change amount is calculated as …**
- **Receipt bar code is designed using …**

Step 5: Interface Type Options

You can gain a high-level view of your story's interfaces by quickly drawing a context diagram of the user story or its higher-level, minimal marketable feature.

Focus on the interface options that are relevant to the

high-priority user role (type and state), actions, and objects (type and state). Then, determine the appropriate type of interface (manual, hardware, system to system, or user).

Our sample user story requires interfaces for the book information (to verify the cost), the cash payment (to secure the payment), and the generated cash receipt.

Select for Value

Together, the product owner and team discuss the options. Based on business and technical factors, they agree on the following:

Book Interface Type Options

Scanner (hardware)
✓ **Keyed in data (UI)**

Cash Payment Interface Type Options

Cash machine (hardware)
✓ **Keyed in data (UI)**

Cash Receipt Interface Type Options

✓ **Printed in store (report)**
Faxed (system to system)
Emailed (system to system)

Step 6: Quality Attribute Options

Quality attributes are "the subset of nonfunctional requirements that describe properties of the software's operation, development, and deployment." [14] Teams sometimes neglect these crucial requirements until well into development. But, we've learned that teams need to explore options for performance, reliability, safety, security, scalability, usability, and so on if they're going to define a ready story and improve the quality of the product's ever-unfolding architecture.

There are several ways to specify quality attributes. For example, you can write them as user stories, use Planguage (a specification language) [15], list the quality attributes as user story acceptance criteria, or incorporate them into the user story text.

One quality attribute we need for our sample story is the response time for printing the receipt. Borrowing from Gilb's Planguage tags, you can specify response-time requirements as follows:

Tag: ResponseTime.CashReceiptPrintLaunch
Scale: Seconds
Meter: Elapsed time between pressing "Receipt" to the start of printing
Minimum: No more than 7 seconds
Plan: 4 seconds
Wish: 2 seconds

Alternatively, you can write your story's quality attributes on the back of the user story card (or in your backlog management tool)—for example, "Cash receipt begins printing within four seconds of pressing the Receipt key."

The Sliced Story

Here's a quick recap. We started with a big, chunky story: "As a *customer*, I want to *buy a product* ..." Using the story-slicing technique, we successively sliced it into these high-value options:

User role type and state: Individual, anonymous buyer
Actions: Verify product price, calculate total purchase amount, secure payment, generate receipt
Objects (type and state): New book, cash payment method, cash receipt
Business rules: Payment currency must be specific to purchase location, payment change amount is calculated as ...
Interfaces:
Book interface type: Keyed in data (UI)
Cash payment interface type: Keyed in data (UI)
Cash receipt interface type: Printed in store (report)
Quality attributes:
Tag: ResponseTime.CashReceiptPrintLaunch ...

Our thinly sliced, ready story meets the INVEST criteria: independent, negotiable, valuable, estimable, sized appropriately, and testable. [16]

In development, the team details each story's selected options. That includes building user acceptance tests, regardless of testing methods or tools—the Given-When-Then construct (e.g., jBehave, easyB, Cucumber), data tables (e.g., FIT or FitNesse), or other scripting tool. Meanwhile, work-ahead analysis continues on lower-priority options for upcoming delivery cycles.

And, yes, we have found this slicing technique useful for requirements in forms other than user stories, including use cases, snippets of text-user requirements statements, events, and features.

Slicing for Success

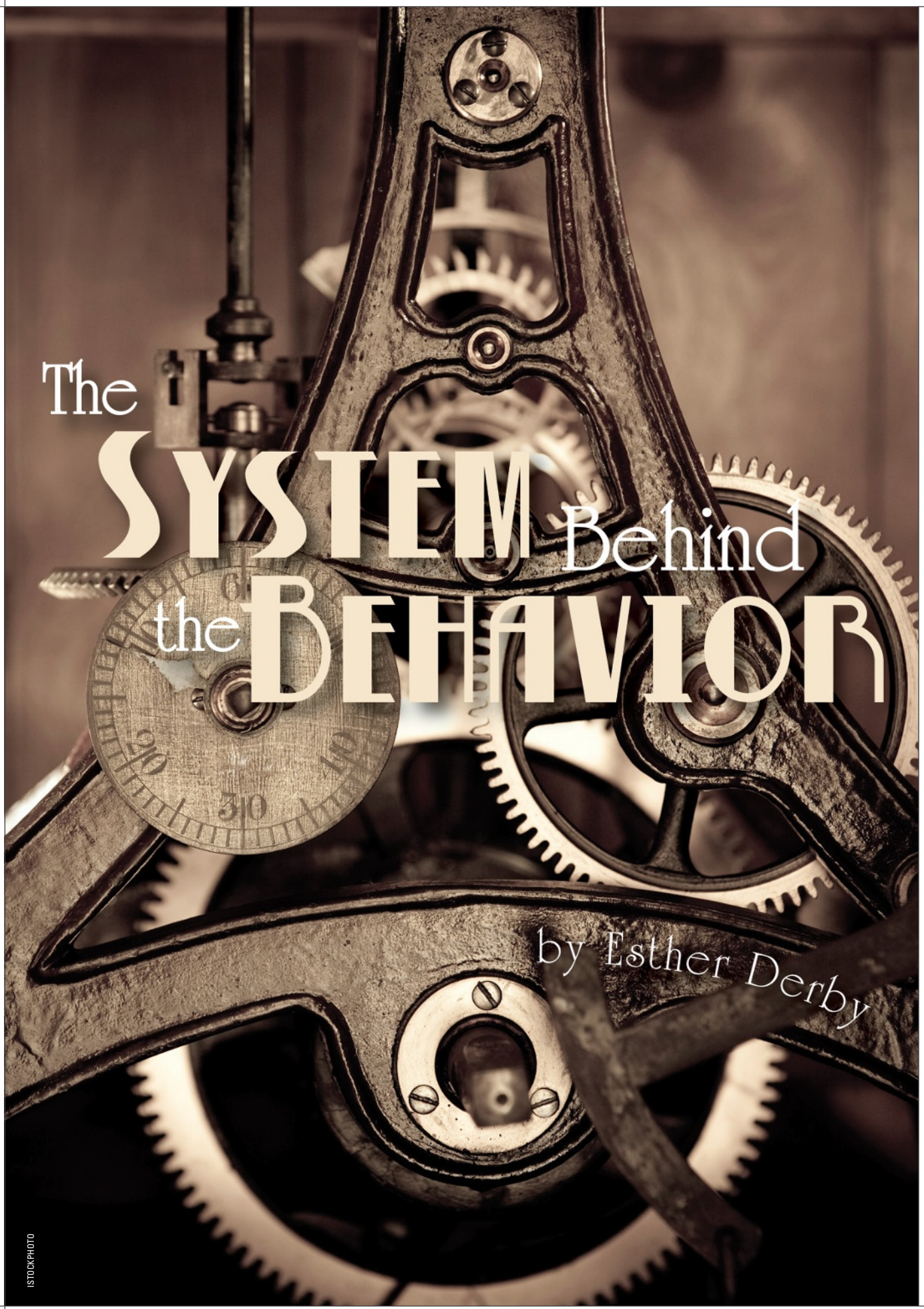
By collaborating with business customers to explore requirements options and successively slice them at the last responsible moment, the team can continually groom the backlog—and continually deliver well-understood, valuable requirements. Along the way, everyone in the project community benefits by expanding and deepening their requirements knowledge.

This "just enough, just in time" slicing method is a fast, efficient, repeatable technique that streamlines planning and jointly engages the customer and team in optimizing value—all goals of successful agile teams. {end}

ellen@ebgconsulting.com
mary@ebgconsulting.com

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

- References



The System Behind the Behavior

by Esther Derby

ISTOCKPHOTO

Sheldon, a development manager, is at the end of his rope with Frank, a developer who works on one of Sheldon's teams. Frank's track record for meeting his own estimates is dismal. He usually takes far longer than he estimates, and he doesn't let his teammates know until it's too late for them to adjust. Frank's poor estimation contributed to the team's missing its iteration goals the past two sprints. Sheldon doesn't expect perfection when it comes to estimates—he knows that it's normal to come in high sometimes and low other times. But Frank is consistently late. The team has tried talking to Frank, but that hasn't changed his behavior. Now, team members are complaining to Sheldon.

Sheldon has done what good managers do—he has given feedback, coached, and provided training. All to no avail. Frank says he's trying, but he's still missing estimates, and his teammates are still annoyed. "I don't want to lose Frank," Sheldon thinks to himself. "He knows more about the domain than anyone on the team. But the way things are going, the team won't benefit from that knowledge. His teammates are so annoyed they hardly talk to Frank anymore."

Sheldon phoned Dawn, a consultant friend, to get some insight on the situation. After hearing the story, Dawn said, "So far, you've looked at this as an individual problem and tried to change Frank. But it might be a system problem—a problem with the way the whole group is working."

Sheldon gave a skeptical harumph. "Seems pretty clear that Frank is the problem."

"Let me share a way to look at the whole team as a system to see if that sheds a different light on the problem," Dawn said.

Framing the Problem

Dawn explained that behind every pattern of behavior, three system factors are at work: containers, differences, and exchanges.

Containers are anything that creates a boundary. Some containers exist in the physical world—for example, cubicles, offices, and meeting rooms. Other containers are psychological but can exert a strong influence on behavior—for example, functional specialties (UI designers, testers, architects) can form strong containers. Timeboxes (the time allocated for a meeting or a task) also can be containers.

Differences are just that—differences that exist within a container or between containers. Differences can be about priority, power, or position. Within a software team (a container), there are differences in technical expertise, domain knowledge, and experience. Differences in national origin, language, gender, and marital status may also influence patterns within a team. Any time you have a group of people, the number of differences is huge. But not all the differences make a difference.

Exchanges are the flow of information, resources, and energy within a container or between containers. We're most accustomed to seeing language-based exchanges, including talking, listening, and reading and writing emails and other documents. Policies, written procedures, and job descriptions are one-way exchanges that influence patterns of behavior across an organization.

Exchanges can take the form of resources, such as money and equipment that support the work. Exchanges can be one way, two way, or multi-directional. They can be full of content and nuance or sparse, conveying little information. Exchanges can have a long lag time before a response or be part of an immediate feedback loop. None of these exchanges is necessarily better or worse; what matters is that the exchanges support a pattern that enables work rather than getting in its way.

You may notice that sometimes something is a container (e.g., a department of functional specialists) and sometimes it's a difference (different functional specialists within a team). Systems are like that—entangled and overlapping, not nice and neat like the boxes on an organization chart. The result often is that when you change one thing, the change will ripple through the system in unexpected ways. A common example involves moving a member of a team to a different part of the office floor (a container change). If the team has gelled, the container change will impact exchanges even when the move is only a matter of yards. Proximity to other groups may inject additional differences as people hear new information or change affiliation. In a system, you can't just change one thing.

After hearing this explanation of the underlying factors that influence behavior, Sheldon was willing to try a new approach. "Maybe I've been missing something by looking at this as a problem with Frank. I really don't want to lose Frank's domain knowledge, so it's worth a try."

Seeing the System

First, Sheldon made a list of what he thought were the most important containers: the team, the planning meeting, the end-of-iteration demo, and the retrospective meeting.

For each of these containers, he listed differences and exchanges, as shown in table 1.

Based on this analysis, Sheldon noticed that while the work was interdependent and he had asked the team to collaborate, the planning meeting didn't support team ownership for completing work. "I'm asking for teamwork and mutual accountability," he thought, "but the fact that people estimate stories individually works against that. So does the fact that people are taking individual responsibility for only the stories they've estimated. They aren't really working as a team. They are still taking individual accountability and assigning individual blame."

Sheldon brainstormed a list of possible actions involving containers, differences, and exchanges, as shown in table 2.

"This is interesting," Sheldon thought as he looked over his list. "Up until now, I've focused on changing Frank, either through talking to him about his personal process or through coaching him on estimating. It looks like there are options for changing Frank's behavior by changing something other than Frank."

Sheldon decided that the planning meeting was the quickest and easiest place to try changing the pattern, since that's where the individual focus on tasks started. He decided he'd ask team members to modify the way they did estimation within the planning meeting.

Container	Differences	Exchanges
Team	Gender Skill level Domain knowledge Time with company Background Technical skills	Day-to-day interactions Questions about product Request for help Team burn-down chart
Planning meeting	Technical skills Domain knowledge Availability Skill level Gender Background Level of participation	Initial description of stories Individual estimating, which is reported back to the group People signing up for the stories they estimated
End-of-iteration demonstration	*Functional roles*: e.g., development team, product team, management *Level in organization*: e.g., staff vs. management *Role in meeting*: story demonstrator, person responsible for accepting the story, meeting facilitator, meeting participant, product stakeholder, interested bystander (not part of the development or product team)	Demonstration of stories completed in the iteration Questions about the stories demonstrated Listing issues with stories Accepting stories Listing what the product owner likes about the product Summary of stories presented Listing of stories not ready to present
Retrospective meeting	Gender Skill level Domain knowledge Time with company Background Technical skills	Individual opinions of what went well, what to do differently Suggestions for action Voting on which action to take

Table 1

Container	Differences	Exchanges
Remove Frank from the team	Raise Frank's skill level through training	Start daily individual reports on estimated time vs. actual time
Pair Frank with another team member for estimating	Raise Frank's skill level through coaching	Pull system for task signup
Estimate as a team, not as individuals		
Pair program		

Table 2

Prior to the next planning meeting, Sheldon asked that three team members work together to break down the stories into tasks and then estimate the task together. The team was willing to try what he suggested.

At first, team members were tentative in discussing stories that they didn't know much about. But after a few minutes, people started asking questions and surprised themselves when they noticed that even questions they thought would be seen as stupid helped the team think differently about the analysis of the tasks needed to complete the story. When they got to the stories that Frank was most likely to work on, everyone held back. Frank started talking through his analysis of the work. Finally, one team member spoke up. "Frank," Jen said, "I don't want to criticize you, but I've noticed that your estimates are usually overly optimistic. I think it's going to take twice as much time to test that database call." Frank looked puzzled and asked, "What am I missing?" Jen walked through her thought process as Frank frantically took notes. It was clear that he'd been missing some important knowledge. "Why don't we work on this together?" Jen suggested.

Sheldon's approach changed the container for team estimating from an individual activity to a small group activity. Changing the container changed the differences—the triad had a more varied pool of technical knowledge and experience than each individual. With a larger container and greater differences, the exchanges broadened and the feedback loop became tighter. One change affected all because they are interconnected.

It's too soon to tell if Frank will become a better estimator, but something in the pattern has already shifted. Everyone on the team is learning more about parts of the system where they haven't worked before. People are starting to learn more about each other's strengths and weaknesses, and they are working together to achieve a common goal.

Not every problem is a system problem. There are still problems that can only be attributed to individual action. But many times, a pattern of poor performance is at least partially attributable to the system. Research by Edwards Deming [1] and others indicates that around 20 percent of the problems are attributable to individuals, while 80 percent are system problems. As managers, we're trained to focus on and assess individual performance. If you believe Deming (and I do), we often use our weakest lever to affect change. But when we learn to think in terms of containers, differences, and exchanges, we have one more tool in our toolbox to address the 80 percent. **{end}**

derby@estherderby.com

Sticky Notes

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

- References
- Further reading

We are excited...

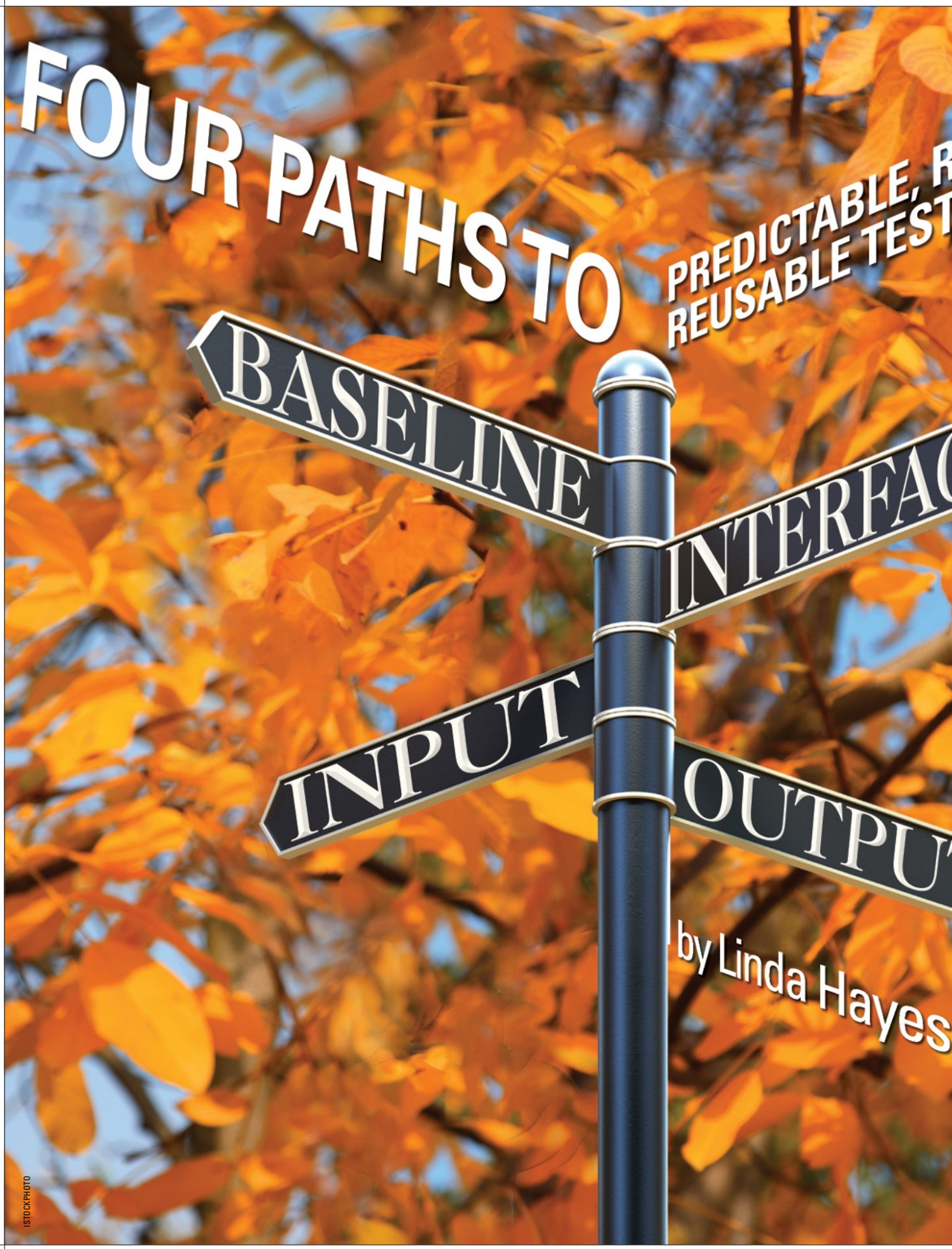
...that Hansoft can handle both schedule driven projects and agile methods like Scrum. The possibility of combining these methods in the same project has really been useful. Hansoft makes our projects more efficient and helps us collaborate much better with partners, customers and suppliers."

General Manager, Space Systems, Swedish Space Corporation.

HANSOFT

Download a free 2-user trial at www.hansoft.se

>> Hansoft is an integrated solution for agile and lean development as well as collaborative Gantt scheduling, real-time reporting, bug tracking / QA, workload coordination, portfolio and document management, used by the most demanding game development studios in more than 20 countries worldwide. Hansoft does not only make team members and managers more productive in their everyday work, it also increases organizational productivity by enabling more efficient production methods and practices. Reduce your project risks with Hansoft, control your success. <<



ISTOCKPHOTO

Designing test data is difficult work. You must understand the requirements, design the test cases, and establish an environment in which test data can be created, maintained, and reused efficiently. However, I've found the rewards can be spectacular: 90 percent productivity improvement for manual testing, increased test coverage, and foundational support for test automation.

Many testers spend the majority of their time locating the data they need for every test case or creating or conditioning the data if it is not available. Test data is typically taken from a copy or subset of production data that is frequently modified, making the contents unpredictable and the test data unreliable. The result is that only a fraction of the possible tests can be executed in the time allocated. However, with an inventory of well-designed and readily available test data, testers can execute orders of magnitude more tests in the same time.

Test data can be organized in four categories, each of which requires a unique creation strategy. If you address each category correctly, your test data will be predictable, repeatable, and reusable.

Baseline

One approach to creating the ideal test environment is to start with a set of empty databases so that all data is loaded under strict control. My experience is that this is not effective. Gone are the days when databases contained data only as numbers and character strings. Today, in addition to that data, databases are complex repositories of procedures, pointers, and rules. Except in the case of an initial implementation of a new application, trying to build a database from scratch for each testing cycle is fraught with difficulty.

The most common source of baseline test data is production data. That data represents reality and is generally available in large volumes. But, production data changes constantly. If the test environment is refreshed every time there is a production database change, then test repeatability is lost.

The best strategy is to start with a copy or subset of production data and then condition it to meet your needs. First, you must assure that no confidential or sensitive information is contained in the test database. Many laws and regulations require that names, addresses, social security numbers, account numbers, medical information—basically anything that can tie a real person to his private information—cannot be revealed to anyone without a need to know (and testers don't need to know). This typically requires scrambling or obfuscating protected fields.

One key to efficiency is to perform this obfuscation process as infrequently as possible. Extract updates of production data only when there is no alternative, because it means you have to repeat the conditioning of your test data.

Next, you must develop a strategy for aging the test data. This means keeping the same relative difference between any stored dates and the system date. This is important because many business rules use time: birth dates, expiration dates, shipping dates, due dates, and so forth. You must either change the system date back or roll the stored dates forward; otherwise, incorrect business rules could be triggered, making the test outcome unpredictable.

So, how do you keep your test data current? By applying changes to the data in small batches but keeping the majority of the contents fixed. Treat your test data baseline like the valuable asset it is by continually adding new content that supports expanding test coverage.

Finally, establish an archive-and-restore procedure to enable you to preserve and later refresh the baseline. You may want more than one version of the baseline to allow you to advance the data to the next state, so testing can continue despite errors. For example, if an update or interface fails, you are able to set the data to the post-update state and continue testing.

Interface

Very few systems operate totally independently. Most either receive data from or send data to other applications—some internal and some external. And, if you have created test data by conditioning production data, you have to keep synchronized any related data from those other systems.

"But even for standard rules, beware. Test data generators can spew out reams of values, but sheer volume is not a virtue. Test cases take time to execute, track, report, and maintain."

You have two options. If the data is received in batch mode, such as a transmitted file, you can condition, archive, and restore it along with your baseline and follow the same aging and other processes for it. The other option is to use a stub or simulation to mimic the interface so that either no data is actually received or you provide simulated, predefined data.

And, don't forget downstream interfaces. The application under test may create data for a downstream system, and it is equally important to maintain synchronization there.

Input

Input data is presented to the system in the usual course of testing. Creating customers, orders, invoices, etc., is typically going to require robust test cases. With proper attention to test case design, these tests can do double duty by seeding the database with useful test data combinations and conditions that will trigger the features and business rules you need to test. Instead of trying to locate a customer who meets certain criteria, just create one through executing a test case. Creating data serves as a useful test and assures that you have a customer with the conditions you need for other tests. Of course, automation may help but, even for manual tests, it is often faster to create the specific data combination you need rather than look for it. Another benefit of creating input data is that refreshing from production does less harm because you are relying on data you add instead of existing data.

It is tempting to automate the generation of this data. Tools are available that enable you to define business as well as standards rules (length, type, limits, etc.) to generate quickly all the combinations needed. But, my experience is that this doesn't work very well. Yes, you can easily write the standard rules, and that may save some time. But it's writing your special business rules—usually in a scripting language—that is lethal. You're effectively duplicating the logic of the system—once for the application and again for the test data creation—and that is a huge and complex effort. (Just ask your developers.)

But even for standard rules, beware. Test data generators can spew out reams of values, but sheer volume is not a virtue. Test cases take time to execute, track, report, and maintain. Less is more when it comes to test data. My advice is to use test data tools to cover the standard rules and help you with optimization techniques like all-pairs, then use application experts to help you manually define the input data that will invoke the business rules.

Finally, make your input data compatible with end-to-end testing. Don't just consider one test case at a time and design the data for it. Instead, look at the entire data flow to make sure that you are creating data that can be reused throughout the end-to-end testing process.

Output

It may seem counterintuitive, but the output data is the hardest to define. The reason is that the output is the expected result of the combination of the baseline, interface, and input data processed by the business rules and reported by the test process. Unlike input data, you can't just make it up. It must accurately and precisely reflect the expected outcome of the test. While some outputs are simple calculations like multiplying units times price or adding a percentage for taxes, often, there are far more complex rules in play that take an expert to understand.

Don't fall into the trap of using the output values from existing production systems. What if those values are wrong? One of the most famous and expensive software defects ever reported lay undiscovered for years in a mortgage underwriting application. The problem was an error in an incredibly complex calculation, and it almost bankrupted the bank. The only way it could have been caught was for an expert to perform the calculation and compare the result to the software. Simply comparing results between test and production masked the error instead of revealing it.

Also, don't buy the idea that you can just rewrite the calculation or formula in a test generator. Unless you have more time and resources than the developers who wrote the application, you can't possibly re-create all of its logic. And, if you can do it faster and more accurately, you should be the developer.

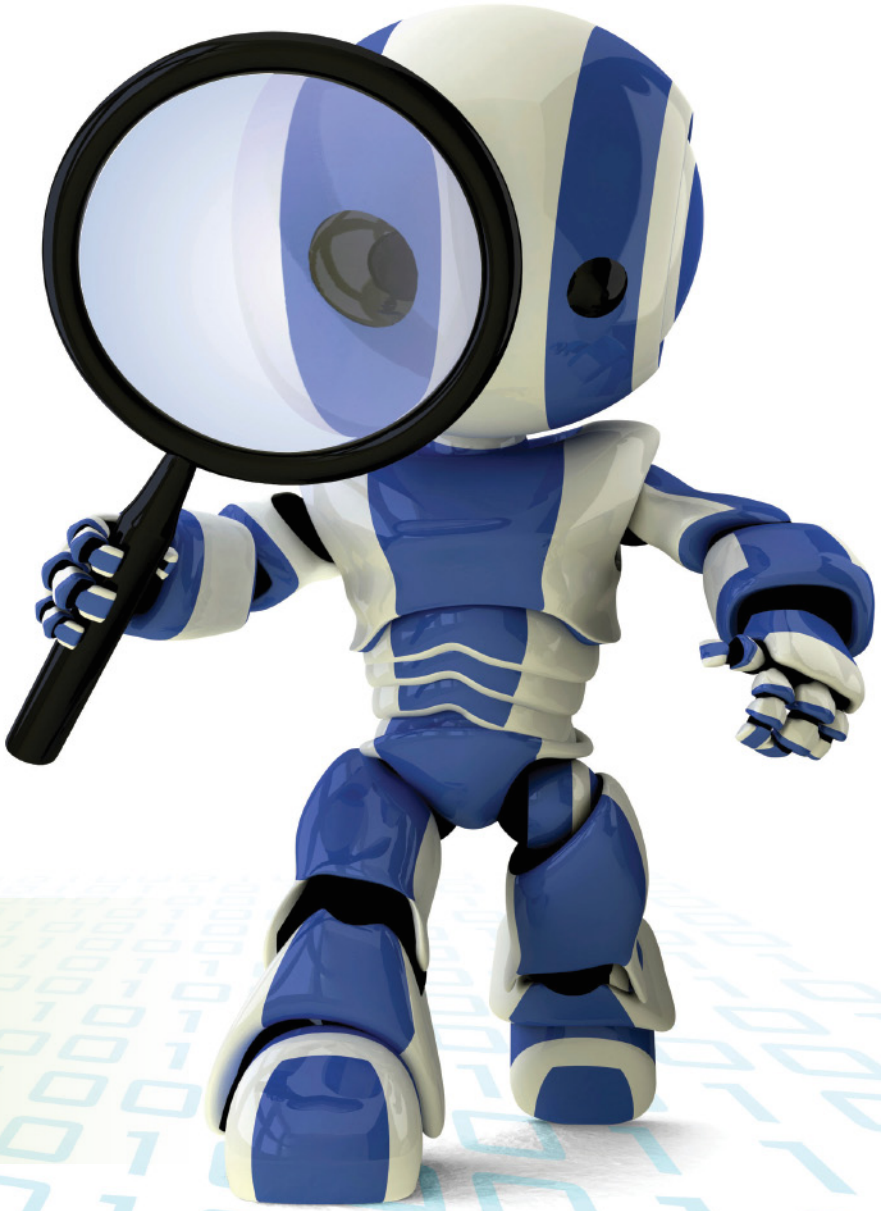
That leaves you with the old-fashioned hard work of taking each and every test case—and each and every set of test data—and walking through the business process with an application expert who can tell you the outcome. I wish there were a shortcut, but there isn't. Like most things, it takes effort to achieve something of value.

Having a well-designed test data environment is the single most valuable asset to a tester, and it is worth every bit of the effort it takes. Done right, it will save enormous amounts of time during test execution, help assure conditions are available to support comprehensive coverage, enable automation to extend coverage and efficiency even more, and—best of all—allow you to validate the critical processes and rules that make the software valuable to the business. {end}

linda@worksoft.com

STAR WEST

SOFTWARE TESTING ANALYSIS & REVIEW



September 26–October 1, 2010

San Diego, California
Hilton San Diego Bayfront

THE GREATEST SOFTWARE TESTING CONFERENCE ON EARTH

CONFERENCE HIGHLIGHTS:

- 100+ learning and networking sessions over six days
- In-depth pre-conference training classes, tutorials, and workshops
- Inspiring keynotes, plus Lightning Strikes the Keynotes
- The New STAR Test Lab launches
- Free Sunday session: Making Agile Work—an Introduction to Agile Software Practices.
- Special events—Grand Casino Night, receptions, bonus sessions
- One-on-One sessions with the STAR presenters
- Testing and Quality Leadership Summit

www.sqe.com/starwest

REGISTER BY AUGUST 27, 2010 AND SAVE UP TO $200.
BRING A COLLEAGUE AND EACH SAVE UP TO $400. BIGGER SAVINGS FOR GROUPS OF 3 OR MORE!

SOFTWARE QUALITY ENGINEERING
CONFERENCES
www.sqe.com

BUILD YOUR CONFERENCE!

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

STAR WEST

SUNDAY

Agile Testing Practices (2 days)
Software Tester Certification—Foundation Level Training (3 days)
Using *Visual Studio® 2010 Ultimate* to Improve Software Quality (3 days)
Bonus Session—Assessing Your Readiness for the ISTQB™ Foundation Exam
Bonus Session—Making Agile Work: An Introduction to Agile Development Practices

MONDAY – TUESDAY

29 In-depth half- and full-day tutorials

Multi-day Training Classes continue

Popular Tutorials Include:

Critical Thinking for Testers
Becoming a Trusted Advisor to Senior Management
Making Test Automation Work in Agile Projects
Free and Cheap Test Tools
Successful Test Automation
A Test Leader's Guide to Going Agile
Essential Test Management and Planning
Exploratory Testing: Now in Session
Measurement and Metrics for Test Managers
Key Test Design Techniques
Adapting to Agile
The Craft of Bug Investigation

WEDNESDAY – THURSDAY

5 Keynote Presentations
41 Concurrent Sessions
The EXPO
Special Events
STAR*WEST* Test Lab
Bonus Sessions
…and More!

Covering topics like:

Test Management
Test Process Improvement
Test Automation
Virtual Test Labs
Embedded & Mobile Testing
Test Techniques
Agile Testing
. . . And Much More

VISIT THE EXPO!

Visit Top Industry Providers Offering the Latest in Testing Solutions

TOOLS • SERVICES • TECHNIQUES • DEMOS

www.sqe.com/starwest

REGISTER EARLY AND SAVE UP TO $200!

KEYNOTES BY INTERNATIONAL EXPERTS

You're Either On the Train or On the Tracks: Radical Realities Shaping Our Future
Linda Hayes, Worksoft, Inc.

Becoming a Great Tester: Inspiration, Perspiration, and Renewal
Isabel Evans, Testing Solutions Group Ltd.

Test Team Leadership—Yes, There's a Place for it in Agile
Bob Galen, iContact

The Human Side of Testing
Thomas McCoy, Australian Department of Families, Housing, Community Services and Indigenous Affairs

Lightning Strikes the Keynotes
Facilitated by Lee Copeland, Software Quality Engineering

"Amazing speakers, great content, great networking, great creative location, great energy, new invigorating ideas."
— Cristy Lucke, QA Lead, Varolii Corporation

FRIDAY

Testing & Quality Leadership Summit

Add a fifth day to your conference event and attend the Testing & Quality Leadership Summit Thursday evening and Friday. Program chairs Ross Collard and Rob Sabourin invite you to join senior leaders from the industry to gain new perspectives and share ideas on today's software testing issues.

Navigating Rough Waters
Jane Fraser, QA Director, Electronic Arts

The Dirty Little Secret of Business
Andy Kaufman, Institute for Leadership Excellence & Development, Inc.

Testing Quality vs. Productivity: A View from the Trenches
Goranka Bjedov, Senior Test Engineer, Google

Think Tank Discussion
Facilitated by Ross Collard and Rob Sabourin

WAYS TO SAVE ON YOUR CONFERENCE REGISTRATION

STAR WEST

Early Bird Offer
Receive up to $200 off the regular conference registration fees if payment is received on or before August 27, 2010.

Bring A Buddy
Bring a colleague and each of you saves up to an additional $200. Any two people registering at the same time save up to an additional $200 off each registration.

Groups of 3 or More Save 25%
Register a group of three or more at the same time and save 25% off each registration. To take advantage of this offer, please call the Client Support Group at 888.268.8770 or 904.278.0524 or email them at sqeinfo@sqe.com and reference promo code GRP3.

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

Alumni Discount
STAR*WEST* alumni receive up to an additional $200 discount off their registration fee.

Multi-day Training Classes + Conference
If you attend any of the training courses and conference, you save an additional $500.

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

THE EXPO *September 29–30*

Visit Top Industry Providers Offering the Latest in Testing Solutions

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

For Sponsor/Exhibitor news and updates, visit **www.sqe.com/STAR*WEST*.**

Industry Sponsors: CONFORMIQ Automated Test Design™ · hp · MICRO FOCUS Leading the Evolution™ · MindTree · telerik deliver more than expected

Media Sponsors: acm queue tomorrow's computing today · BETTER SOFTWARE MAGAZINE · CoDe www.code-magazine.com · IT Today · Methods & Tools · SearchSoftwareQuality.com · StickyMinds.com · Toolbox.com

www.sqe.com/starwest REGISTER EARLY AND SAVE UP TO $200!

LoadLabs

DENVER, CO—ProtoTest has announced the launch of LoadLabs—the first and only flat-rate load and performance testing solution. LoadLabs represents a completely new way to think about load and performance testing.

LoadLabs takes the hassle of load testing off your plate and helps deliver improved online app and Web site performance in the most cost-effective way possible. Other load test services offer price "estimates," which create a significant barrier to determining the cost effectiveness of a test. LoadLabs eliminates that variable.

Tests are done in about a week, and reports provide both top-line and highly detailed information. Retests are also included, so you can make sure your sites and apps work the way they are supposed to—before they go live.

For more information, visit loadlabs.com.

GUI*dancer* 4.1

BRAUNSCHWEIG, GERMANY—Version 4.1 of GUI*dancer* is available for immediate download. This service release includes better support for working with very large test projects and has the added functionality of automatically taking screenshots when errors occur in tests. The new version also sees the Mac OS X support move from beta to fully supported. GUI*dancer* is an Eclipse-based testing tool that allows modular and agile acceptance testing for Swing, SWT/RCP, and HTML applications.

GUI*dancer's* new feature automatically takes a screenshot when an error occurs in a test, reducing the effort required to analyze test failures and to document errors and their causes. Combined with the considerable extensions to test result reports in version 4.0, GUI*dancer* now provides the option of not only viewing a screenshot directly in the client but also of adding the screenshot to a BIRT report of the test run. Whether developing new tests or analyzing existing ones, this new screenshot function means that testers can quickly analyze errors occurring in their tests.

GUI*dancer* 4.1 includes full support for the Mac OS X platform. The GUI*dancer* client can be used on Mac systems, and Swing and HTML applications (Firefox and Safari) can be tested on the Mac OS X platform. This addition allows testing of applications on Windows, Linux, and Mac machines for cross-platform peace of mind in acceptance testing.

For more information, visit bredexsw.com.

EggPlant for Linux

LONDON, UK—TestPlant has updated eggPlant for Linux with release of version 10.2. Improvements include better GUI and Script Editor performance, and new SenseTalk commands. Engineers will notice reduced screen updates requested from the VNC server, improving connection performance with slower servers and on slower connections. This improvement will significantly help multi-located QA and development teams work proactively, reducing time to market.

The new release features a new TypeText recording feature that allows you to automatically capture most keyboard input by just typing. New SenseTalk features expand eggPlant's text handling and provide more options for value comparisons. Improved control over helper suites and enhancements to the case and bug submission panel allow version 10.2 of this universal GUI test tool to bring with it key features that reduce the learning and deployment curve of the tool, thus significantly helping QA engineers who are new to automation.

The updates to the Linux version strengthen eggPlant's non-invasive architecture, clever image recognition technology, and ability to deliver test results using less resource commitment.

For a trial, visit testplant.com/products/eggplant_functional_tester/trial.

Copyright © 2010, Variad Corporation. All rights reserved.

$0
TECH. SUPPORT

WE HAVE THE BEST NERDS

TO ANSWER ALL YOUR SUPPORT QUESTIONS. OH, YEAH. FOR FREE!

- BUG TRACKING
- REQUIREMENTS
- TEST CASES
- TASK TRACKING
- FREE TECH. SUPPORT
  included with every BugHost subscription

THE RIGHT TOOL. THE RIGHT PRICE. THE ORIGINAL

BugHost™

www.BugHost.com

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

FAQ

expert answers to frequently asked questions

What can I do to estimate testing activities more accurately?

by Dawn Haynes
dhaynes@perftestplus.com

I never liked being asked to estimate how long it will take to test something. Doesn't everybody know that it will take until the sun explodes to test software to the satisfaction of a tester? Besides, no matter how lean my estimates were, the team always chopped them in half to fit them in the schedule! After a while, I started to think asking for testing estimates was something between a trick question and a bad joke.

Finally, I decided that I must not even understand the question, so I started asking other folks how they estimated. Surprisingly the challenges were common, but I found most folks had a scheme accurate enough to make estimating less painful. Their techniques ranged from cynical (Ask how long until it ships, then use the answer as your estimate) to math-magical (Come up with what you think is an accurate number, then double it and add 3) and everything in between.

In every course I teach I get asked about better estimation. Compiled from much research, advice, and experience, here's the set of working ideas that I share with folks striving to improve:

Don't estimate fantasy: If a target ship date is established, don't estimate more time than is available between now and then. First, you're highly unlikely to get a schedule extension based on your estimate. Second, this is your opportunity to change the question to "What testing do you think you can accomplish before the ship date?" That can lead to a conversation about what testing is most critical and sometimes even get you that extension you wanted in the first place. Remember, your primary mission is to test what the stakeholders need you to test—not to test everything you want to test.

Learn about your projects: Identify things germane to your projects that you should consider in your estimates. Special tasks or activities (reviews, configuring test environments, creating test data), notable overhead items (meetings, activity logging, status reporting), even tendencies (scope creep, changing requirements, late deliveries of software builds to test, blocking bugs, buggy builds, broken builds). Note these things and state that they can severely impact the accuracy of your estimates.

Learn about yourself: Monitor your tasks and activities carefully for a while. Be sure to track all the things you do, don't do, and why. Track approximately how long you invest in these things. Use these measures to figure out if they are worth doing on future projects and, if so, how long they might take. Consider time spent in meetings, email, mentoring others, and similar activities. Don't forget to estimate the time spent learning from people or exploring the software or system prior to testing!

Post-process: Look to improve your estimates by looking back across your projects. Compare your actual efforts to your early estimates and tune future estimates accordingly. In the long term, remeasure and recalibrate continuously. Oh yeah, and don't forget to allocate time for this in your estimate!

So, You Want to be a Consultant?

Many practitioners see "consultant" as their ultimate career goal. But what does that really mean? Fiona Charles explains in this email response.

by Fiona Charles | www.quality-intelligence.com

Hi Ben,

Here's some follow-up on our quick conversation yesterday. You asked what I thought you should do next to prepare yourself to be a consultant.

Did you notice how you asked the question? You began by saying, "I want to be a consultant like X." I've since wondered what you were thinking. Do you know what kind of consultant X is? Do you know what kind of consultant you want to be?

I don't mean what you want to consult about, though that's obviously basic. But it's essential to know how you want to conduct yourself as a consultant in relation to your clients. How do you see yourself in a consulting role? What kinds of organizations and people do you want to work with? Do you want to help your clients or influence them—and do you see any difference? What kind of help do you want to offer? What kind of influence do you want to have?

In my experience, there are two principal types of consultants. At one end of the spectrum is the group that is primarily prescriptive. These are the consultants who come in with a set of preconceived answers and ideas about the right way to do things. The big consulting firms tend to be like this, with a belief in industry best practices and packaged intellectual property intended to implement them. So do some well-known individual consultants who identify themselves with particular methods.

At the other end is the consultant who comes in knowing how to approach problems but has an open mind about what the particular issues and solutions might be for a client. This is my preferred style of consulting. I may well have ideas about the best way to do lots of things, but I try to put the customer's needs first. Quite often, the customer knows the best solution for them, and it might not fit with my views about how testing (or software development or management or anything else) should be done. I try to understand their context and the reasons for their established practices, and I try to see what's valuable about what they are already doing. I also work to open up the customer to other possibilities, but as much as possible, I collaborate with them. I ask a lot of questions and listen to the answers. I try to learn and talk their language, rather than trying to get them to talk mine.

I have pretty strong views on consulting. I think empathy with the customer is central to effective consulting, and I don't look down on customers because they don't think about things the way I do. After all, I learn from them, too. (Although, if there's a major disconnect, I might decide they'd be happier with a different consultant.)

But, although it's foreign to me, the prescriptive consulting model is legitimate, too—and it's what a lot of clients prefer. Many organizations feel more comfortable with a consultant who can give immediate answers and help them implement "best practices." Or they choose a consultant who is known for promoting a particular methodology.

> "I think empathy with the customer is central to effective consulting, and I don't look down on customers because they don't think about things the way I do."

So where do you want to be on that consulting style spectrum? The answer will shape your eventual consulting practice and your customer base. If you want to be at the prescriptive end, then you'll probably end up working mostly with clients who already think the way you do. If you want to be more open, then you'll have more opportunities to work with a broader range of customers and possibly have a really positive influence on people you might not otherwise reach.

I've talked about the advantages of working for a consulting firm. It's only one of many ways to begin a consulting career, though, as it was my path, it's the one I'm best qualified to talk about. If you don't already know your preferred consulting style, this can be an opportunity to explore the spectrum. You will see examples of behavior that you want to emulate in your own practice and also of behavior you want to avoid.

A consulting firm will teach you how to act like a consultant and various other skills that will be useful to you later in your own business, including: how to write a proposal, how to sell—yourself and your services, how to write a professional consulting report that will engage management attention, and how to start a gig and how to wrap one up. You will have frequent interviews with potential clients and probably

become very good at them. It will be a course in a different kind of professionalism from that appropriate to an employee.

A less obvious benefit is the opportunity to try out the "consultant" lifestyle in a relatively safe environment. In my first consulting company, the branch manager had a saying: "If you're still here in three years, you're a consultant." That was because about two-thirds of our new hires bailed out in the first three years, often shortly after their first project. Even on delivery projects, consultants have to be prepared to go somewhere new every few months, perhaps every few weeks. Not everyone has the stomach for constant change and uncertainty. Consultants need to be comfortable with never knowing where they're going to be working next, what they're going to be doing, with what technology, and with whom. They also need to be philosophical about not always being allowed to finish a job. Better to try out your comfort level with those things when someone is paying you a regular salary than when you're on your own and also worrying about whether you're going to eat next week!

Large firms will give you access to larger client organizations and projects but are typically prescriptive—important to keep in mind if that's not where you want to go. Small firms may give you more scope to try out different roles, but large or small, if you decide to join a consulting firm, I recommend starting in its delivery practice rather than the consulting practice. This would mean working on software delivery projects at client sites. It would give you a really wide experience of a variety of companies and industries and an on-the-ground feeling for how they do software development and testing. It would help you understand their points of view—what matters to their businesses, how they think about risk, etc. Over time, you would build a well of experience from which to draw later as a consultant. And you'd probably also have opportunities to take on consulting projects with people from the firm's consulting practice guiding you.

Of course, there are also books you should read. Jerry Weinberg's *Secrets of Consulting* [1] and *More Secrets* [2] top my list. Peter Block's books are also excellent; you could start with *Flawless Consulting*. [3]

Most importantly, Ben, picture yourself ten or twenty years from now. Who is the consultant you want to be?

Best regards, Fiona **{end}**

This article originally appeared on StickyMinds.com.

Sticky Notes

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

- References
- Further reading

index to advertisers

ADP East 2010	www.sqe.com/adpeast	Inside Back Cover
BugHost	www.BugHost.com	33
Hansoft	www.hansoft.com	15
Hansoft	www.hansoft.com	25
Hewlett-Packard	www.hp.com/go/agile	Back Cover
McCabe	www.mccabe.com	11
Microsoft	www.microsoft.com/visualstudio/test	8
Original Software	www.OriginalSoftware.com	Opposite Page 2
Rally Software	www.rallydev.com/bsm	Inside Front Cover
Ranorex	www.ranorex.com	5
Seapine	www.seapine.com	1
SQE Training—Training Week	www.sqetraining.com/go?TrainingWeeks	12
SQE Training—Live Virtual Training	www.sqetraining.com/go?LiveVirtual	14
STAR*WEST* 2010	www.sqe.com/STARWEST	29–32
TechExcel	www.techexcel.com	2

Display Advertising
advertisingsales@sqe.com

All Other Inquiries
info@bettersoftware.com

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

NOVEMBER 14–19, 2010

ORLANDO, FLORIDA

AGILE DEVELOPMENT PRACTICES CONFERENCE East

Conference Sponsor:

RALLY SOFTWARE

www.sqe.com/adpeast

Choose from a full week of learning:

SUNDAY
Multi-day Training Classes begin

MONDAY–TUESDAY
In-depth Tutorials In half- and full-day formats

WEDNESDAY–THURSDAY
Keynotes, Concurrent Sessions, EXPO, Receptions, Bonus Sessions, and more

FRIDAY
Agile Leadership Summit

REGISTER BY SEPTEMBER 17, 2010 AND

SAVE UP TO $400

GROUPS OF 2 SAVE EVEN MORE!

SOFTWARE QUALITY ENGINEERING CONFERENCES

100% OF 2009 ATTENDEES RECOMMEND THE AGILE DEVELOPMENT PRACTICES CONFERENCE

HARNESS

the power of Agile.

HP Application Lifecycle Management for Agile initiatives

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

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

Outcomes that matter.

Get more information at
hp.com/go/agile