May/June 2011 $9.95 www.StickyMinds.com

BETTER SOFTWARE™

A TECHWELL PUBLICATION

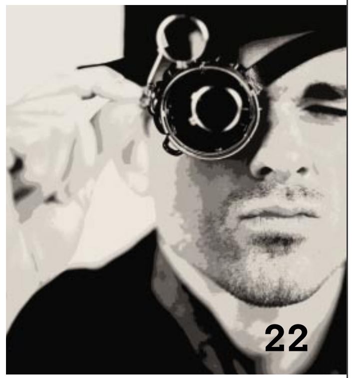
AGILE BALANCE
Planning & analysis working together

MAP YOUR JOURNEY
Exploratory test documentation

MAKE YOUR TELE-TEAM WORK

BETTER SOFTWARE™
A TECHWELL PUBLICATION
Volume 13, Issue 3 • MAY/JUNE 2011

CONTENTS



BETTER SOFTWARE™
A TECHWELL PUBLICATION

Publisher
Software Quality Engineering, Inc.

President/CEO
Drew Thoeni

Vice President of Publishing
Holly N. Bourquin

Editor in Chief
Heather Shanholtzer

Editorial

Managing Technical Editor
Lee Copeland

Online Editor
Joseph McAllister

Production Coordinator
Cheryl M. Burke

Design

Creative Director
Catherine J. Clinger

Advertising

Director of Sales
Sonia Lavin

Customer Success Manager
April Evans

Circulation and Marketing

Circulation Coordinator
Jamie Green-Gago

Product Marketing Manager
Diara A. Sullivan

CONTENTS

columns

in every issue

MARK YOUR CALENDAR

SQE TRAINING
www.sqetraining.com

software tester certification

www.sqetraining.com/certification

June 14–16, 2011
Ft. Lauderdale, FL

June 14–16, 2011
Stamford, CT

August 23–25, 2011
San Jose, CA

August 23–25, 2011
Boston, MA

August 30–September 1, 2011
Charlotte, NC

training weeks

www.sqetraining.com/testing

Software Testing Training Weeks
September 19–23, 2011
Washington, DC

October 17–21, 2011
San Francisco, CA

November 14–18, 2011
Tampa, FL

conferences

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

STAR*WEST* 2011
Software Testing Analysis & Review
www.sqe.com/starwest
October 2–7, 2011
Disneyland Hotel
Anaheim, CA

Agile Development Practices East
www.sqe.com/adpeast
November 6–11, 2011
The Rosen Centre
Orlando, FL

STAR*EAST* 2011
Software Testing Analysis & Review
www.sqe.com/stareast
April 15-20, 2012
Rosen Shingle Creek
Orlando, FL

BETTER SOFTWARE™
A TECHWELL PUBLICATION

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.

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.
340 Corporate Way, Suite 300

PRINTED WITH SOY INK

Contributors

Laura Brandenburg is a business analyst consultant, author, and mentor. She has ten years of experience across technology leadership, analysis, project management, and QA. Laura hosts *Bridging the Gap*, a blog helping business analysts advance their careers, and is the founding instructor of My Business Analysis Career. She authored "How to Start a Business Analyst Career" and "The Promotable Business Analyst" series. Visit Laura's blog at www.bridging-the-gap.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.

Lisa Crispin is the co-author (with Janet Gregory) of *Agile Testing: A Practical Guide for Testers and Agile Teams* and a contributor to *Beautiful Testing*. A tester on agile teams for the past ten years, Lisa enjoys sharing her experiences at conferences and user group meetings around the world. For more about Lisa's work, visit www.lisacrispin.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.

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.

Contributors

Don Gray's goal to answer "What is the earliest indicator for a project's status?" has led him to focus on such diverse topics as communication, personality types, team styles, systems thinking, and human systems dynamics. Don's varied interests and client experience provide a platform for helping clients introduce and work through change as they transition to agile development practices. His blog can be found at www.donaldegray.com.

For more than ten years, David Hussman has evolved the successful adoption and use of agile methods. He believes in helping companies design processes that help them instead of simply teaching them a process to follow. When he is not coaching or teaching, you can find David speaking at conferences around the world and publishing to a variety of sources. He has contributed to several books as well as helped to create agile curriculum for The University of Minnesota and Capella University. For more information about David or his company, DevJam, check out the DevJam website devjam.com.

Jonathan Kohl is a software testing consultant with Kohl Concepts Inc., based in Calgary, Alberta, Canada. Jonathan writes about and speaks on software testing. Read more of his work at www.kohl.ca. Contact Jonathan at jonathan@kohl.ca.

Nanda Lankalapalli is an architect, Certified Scrum Practitioner, and agile coach. He built and led high-performance teams and is an active contributor to the agile community. Nanda is one of the founders of Agile Hyderabad user group and has spoken at several agile conferences organized in Hyderabad, India. Nanda is part of Eplan Services and telecommutes from India.

Filip Szymanski is director of products at HP Software and is responsible for the HP Application Lifecycle Management product line. He leads the team responsible for current and future application lifecycle management releases, including requirements, test, and defect management. Before joining HP via the Mercury Interactive acquisition, Filip was in technical sales where he enabled the sales organization to sell higher revenue products and services.

CM CROSSROADS
A TECHWELL™ COMMUNITY

AGILE JOURNAL™
A TECHWELL™ COMMUNITY

EXCITING NEWS FOR *BETTER SOFTWARE* REA

We've added CM Crossroads and Agile Journ

Software Quality Engineering, Inc. (the publisher of *Better S*
Journal web communities, including all online publications,

What this means for our *Better Software* magazine subscri
on, but we've added new voices, new viewpoints, and new
management, project management, software testing, and r

About CM Crossroads and Agile Journal

Since 1998, CM Crossroads has been the web's most comp
application lifecycle management. CM Crossroads offers in
center and jobs board, and an extensive library of resources
edge and is the publisher of the Configuration Managemen

Agile Journal is the premier social media community for ag
for an agile business. Agile Journal delivers thought leader
from hands-on developers and project managers. The Agile
and openly in the discussion forums and in member blogs.
articles from some of the industry's innovators and thought

Visit cmcrossroads.com and agilejournal.com to find out more abo preconfigured images with the required set of software simulate thousands of users on the application under test, instantaneously. Security concerns around systems being exposed through the cloud are done away with extending one's own infrastructure through a Virtual Private Cloud.

At QA InfoTech, JMeter has been engineered for better reporting and monitoring with the expertise of the in-house R&D team. The extended JMeter incorporates capabilities of Jasper and Crystal Reporting APIs' for efficient and detailed reporting of the load test results and addition of SNMP monitoring plug-in. This when clubbed with the capability of cloud computing for load generation has proved to be an effective performance testing solution for products and customers from all over the globe. Measurable ROI from such a solution include: Doing away with expensive licenses, commercial hardware load generators and 24X7 hardware maintenance and support.

This engineering technique at QA InfoTech has helped our customers' performance tune some of their most complex systems such as conferencing products with heavy audio and video streaming over secured protocols. Other notable achievements include: real world traffic simulation from different geographies, performance engineering on products developed in an agile life cycle, optimized performance testing efforts on prioritized combination of operating systems and browsers, all of which have been done at significantly lower costs compared to use of commercial tools.

QA InfoTech is an Independent software testing firm headquartered in India with offices in the US.

For more details, case studies and other performance engineering methodologies, we can be reached at info@qainfotech.com

Sample Report generated

Load Test 1 : 1000 VUsers : Pre-Prod Environment

In Conclusion

- Load test was declared failed as response times for login and launch were greater than 30 seconds for 90% of requests.
- However, User actions and document conversion was below 3 seconds.
- We have also verified the response time for login, manually, which was in the ballpark of 20 to 30 seconds.

Test Result	Fail

Groups	Response Time Criteria(Sec)
Launch	30.365
Login	32.345
User Actions	2.455
Document Conversion	1.233
Logout	0.233

Pre-Prod 1000 VUsers Load Test Run1
Date: August 27, 2010

Load Runs Information:

- **Ramp UP Information:** 1 VUser per 1 Second(s)
- **Total Number of VUsers:** 1000
- **Duration:** 1 hour(s)
- **Start Time:** 15:00 hrs IST
- **Total Duration:** 30 minutes (excluding ramp up/ ramp down)

Scenario Covered: Please find below the VUser distribution of moderator and participant executing each scenario

Functionality	No. of Moderators (Conference)	No. of Participants	Total Strength
DocumentShare	14	326	340
Invite	114	NA	114
AppShare	5	125	130
Chat	8	200	208
Annotation	8	225	208

Legend	Optimum time values for different user experiences while executing various **Functionalities in the Application**
Load Test Pass/ Perfect User Experience	90% of the samples get a response time of **<=5 seconds**
Load Test Pass with Few Exceptions/ Ok User Experience	90% of the samples get a response time of **> 5 and <=12 seconds** **OR** 30% of the samples get a response time of **> 9 seconds**
Load Test Fail/ Poor User Experience	90% of the samples get a response time of **> 12 seconds** **OR** 50% of the samples get a response time of **> 9 seconds**

Figure: Legend which demarcates Test Results

CLOSE WHITE PAPER

Spec

DERS

al to our library of resources to help you build bettter software!

Software magazine) has purchased all assets associated with the CM Crossroads and Agile virtual expos, web seminars, and content associated with both websites.

bers: You will have access to the high-quality content and expert opinions you've come to rely outlets for learning. Our technical coverage now includes agile development, configuration elated topics on career development, management, and leadership for IT professionals.

rehensive developer community and resource center focused on configuration management and teractive discussion forums, a broad range of original articles and newsletters, a career search s in the CM Yellow Pages. CM Crossroads hosts the Configuration Management Body of Knowl- t Journal.

ile, focused on providing members the information and resources they need to develop software ship and pragmatic advice from a wide range of industry experts, as well as direct feedback Journal community enables members to interact with their peers and share ideas on agile freely The media center offers a wide range of educational videos, weekly webcasts, whitepapers, and leaders. Agile Journal is the publisher of the Agile Journal online magazine.

ut these communities and to subscribe to CM Journal or Agile Journal.

CM CROSSROADS™
A TECHWELL™ COMMUNITY

AGILE JOURNAL™
A TECHWELL™ COMMUNITY

Exciting News for *Better Software* Readers

We've added CM Crossroads and Agile Journal to our library of resources to help you build bettter software!

Software Quality Engineering, Inc. (the publisher of *Better Software* magazine) has purchased all assets associated with the CM Crossroads and Agile Journal web communities, including all online publications, virtual expos, web seminars, and content associated with both websites.

What this means for our *Better Software* magazine subscribers: You will have access to the high-quality content and expert opinions you've come to rely on, but we've added new voices, new viewpoints, and new outlets for learning. Our technical coverage now includes agile development, configuration management, project management, software testing, and related topics on career development, management, and leadership for IT professionals.

About CM Crossroads and Agile Journal

Since 1998, CM Crossroads has been the web's most comprehensive developer community and resource center focused on configuration management and application lifecycle management. CM Crossroads offers interactive discussion forums, a broad range of original articles and newsletters, a career search center and jobs board, and an extensive library of resources in the CM Yellow Pages. CM Crossroads hosts the Configuration Management Body of Knowledge and is the publisher of the Configuration Management Journal.

Agile Journal is the premier social media community for agile, focused on providing members the information and resources they need to develop software for an agile business. Agile Journal delivers thought leadership and pragmatic advice from a wide range of industry experts, as well as direct feedback from hands-on developers and project managers. The Agile Journal community enables members to interact with their peers and share ideas on agile freely and openly in the discussion forums and in member blogs. The media center offers a wide range of educational videos, weekly webcasts, whitepapers, and articles from some of the industry's innovators and thought leaders. Agile Journal is the publisher of the Agile Journal online magazine.

Visit **cmcrossroads.com** and **agilejournal.com** to find out more about these communities and to subscribe to CM Journal or Agile Journal.

Long-distance Relationships

I've just returned from working the STAREAST conference in Orlando—an event I look forward to all year because I use a few hours of the week to sit down with the rest of the editorial team and begin planning content for the next year. This might not seem like an event worth mentioning; people sit down with their teammates all the time. The significant part of this meeting is that two of the three editors live and work far from the home office, so it's always fun to meet in person.

As a member of a distributed team with very flexible work hours, I am always looking for tools and tips for facilitating communication among team members. To my delight, this issue's cover story could not have been more helpful if I had commissioned Lisa and Nanda to write it just for me.

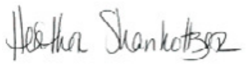
"Make Your Tele-teams Work" is a real-life account of how one distributed development team shortens the distance between its members using a variety of high- and low-tech tools—one of which, "Virtual Nanda," I find especially to be a stroke of genius. In this article, Lisa and Nanda address specific challenges programmers and testers face on virtual teams and offer helpful tips for a productive and healthy working relationship.

Also in this issue, Jonathan Kohl dispels some myths regarding documentation requirements on exploratory testing projects, which will enable you to explore away, even on highly regulated software. In "Documenting Your Exploratory Testing," Jonathan describes several documentation methods that translate well to these types of projects, including guidance documents to help focus testing, test coverage reports, and recording what you've tested.

Perennial favorites Ellen Gottesdiener and Mary Gorman return this issue with an enlightening look at the relationship between planning and analysis on agile projects. When done hand in glove, these activities work together to groom your backlog and maximize business value.

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

Happy reading,



Heather Shanholtzer
hshanholtzer@sqe.com

Works as Designed

In the wake of Japan's recent nuclear emergencies, the weakness of the "works as designed" argument has again been exposed.

by Lee Copeland | lcopeland@sqe.com

On Friday, March 11, 2011, a magnitude 9.0 earthquake struck northern Japan, triggering a tsunami that caused widespread devastation and crippled the Daiichi power station, which comprises six nuclear reactors. In the week following the earthquake, Japan experienced 262 aftershocks of magnitude 5.0 or greater. That's a record number of aftershocks for any quake in Japanese history. During that week, neither Tokyo Electric Power Company nor the Japanese government was able to bring the damaged reactor under control.

Throughout the world, debates have again become heated about the safety of the nuclear generation of electricity. In an op-ed piece for CNN.com, Senator Lisa Murkowski (R-Alaska) defended nuclear power: "Those concerned that the Daiichi power plant will become another Chernobyl must acknowledge that when the earthquake hit Japan, the nuclear reactors all automatically shut down as designed. Within seconds, the control rods were inserted into the reactor's core, the nuclear chain reaction stopped, and power levels dropped to about 7 percent of full power." She continued, "We should also keep in mind that a forty-year-old nuclear power plant withstood a magnitude 9.0 earthquake and shut down exactly as it was designed to do. The backup generators kicked in as they were supposed to, and even when a larger-than-anticipated tsunami flooded the generator, the reactor operators switched to emergency battery power, which lasted for eight hours [1]."

> "Even though the software is difficult to use, produces incorrect results, has some malicious side effect, or in some other way is not fit for use, it seems that "works as designed" is an acceptable response."

The fact that nuclear fuel is still exposed, is not under control, and is spewing forth radiation is apparently not important to Murkowski. What is important is that the reactors did what they were designed to do in a specific set of circumstances. Senator Murkowski has fallen into the classic "works as designed" fallacy. She emphasized that "the nuclear reactors all automatically shut down *as designed*" and the power plant "shut down exactly as it was *designed to do*." In a very narrow, technical sense, Murkowski is correct. However, like one of Saxe's blind men describing an elephant [2], she has focused on one aspect of the situation while ignoring other, perhaps fatal, aspects. At the end of the article, CNN.com noted: "The opinions expressed in this commentary are solely those of Lisa Murkowski." Unfortunately, this is not the case.

How many times have we heard the phrase "works as designed" used in software development? Even though the software is difficult to use, produces incorrect results, has some malicious side effect, or in some other way is not fit for use, it seems that "works as designed" is an acceptable response. Well, for professionals, it isn't acceptable. Please understand, I'm not comparing the magnitude of Japan's triple national disasters with our problems in software development. However, I am concerned that the "works as designed" fallacy is once again being offered as a legitimate argument.

The purpose of any software system is to provide value to people by helping to solve their problems or lighten their loads. Users should expect that software helps them, not harms them. "Works as needed," "works as advertised," and just "works" should be our goal—not "works really well in this situation but otherwise is detrimental to the user."

In his *Better Software* magazine article "The Myth of Risk Management" [3], Pete McBreen writes, "Risk management is an illusion … Projects run into problems because they are hit by something that everyone knew about but was unwilling to talk about, something totally unexpected, something that was rated as highly unlikely, or something that was made even worse by the 'mitigation strategy.'" Pete adds, "When the people in charge of managing the risk are the ones creating the risk, it is not hard to see that risk management is an illusion."

In the next few years there will undoubtedly be dozens of inquiries into the Daiichi disaster. Each will attempt to fix the blame. Perhaps a few will actually attempt to fix the problem. My guess is that most officials will praise those parts of the systems that worked as designed while missing the bigger picture. Let's stop doing that in our industry. {end}

Sticky Notes

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

- References

Business Analysis Career Paths

In the past, business analysis was primarily a career track to project management. Today, your path could go many ways.

by Laura Brandenburg | laura@clearspringanalysis.com

Many talented professionals find themselves successful in their business analysis careers and start looking for what is next on the horizon. Others consider careers in business analysis and wonder what their career prospects are.

Five or ten years ago, business analysis was often heralded as an entry-level path into project management. Today, the path from business analysis to project management is still a legitimate career direction, but we are seeing a reverse trend—project management professionals transitioning into business analysis careers.

If Not Project Management, Then What?

Senior business analyst job titles are doled out by individual organizations, and there is great variety in what qualifications constitute a senior-level individual. Sometimes, several years of experience will qualify one to be a senior business analyst. But in most organizations, senior-level qualifications include a combination of business analysis capabilities, specific expertise (such as domain, industry, or technical expertise) valued by the organization, and strong soft skills, such as communication and relationship building.

More generally, within the profession of business analysis, senior BAs are those who either have or would qualify for IIBA's CBAP designation. This means they have five years of working experience across a wide collection of knowledge areas. We call these professionals generalists. They are professionals you could assign to almost any project in any domain and be confident they would find the path to success.

Beyond advanced generalist roles, there are several career directions a senior-level business analyst might choose to take. These paths are not mutually exclusive, and, in fact, many business analysts blend aspects from multiple roles.

Let's consider five of the more common and enticing paths for senior business analysis professionals.

> "Beyond advanced generalist roles, there are several career directions a senior-level business analyst might choose to take."

Enterprise Analysis

Enterprise analysis is a knowledge area from the *Business Analysis Body of Knowledge Guide* and includes the understanding and analysis required to develop a solid business case. If you ask "why" or define the problem you are trying to solve within the context of a specific project, you are doing enterprise analysis.

As a senior-level responsibility, enterprise analysis can take the form of planning and scoping several interrelated initiatives and helping senior-level business stakeholders make informed decisions about which initiatives to tackle and how to tackle them.

An emerging career path for senior business analysts is to focus almost solely on enterprise analysis activities. These analysts proactively work with the business to understand how it is organized and what its pain points are and to support the creation of project proposals and business cases for new projects. Enterprise analysts work "before the project" to ensure that the organization as a whole is making the best possible decisions about how to invest resources to achieve its goals or execute its strategy.

Community of Practice Leader

As business analysis takes hold as a profession, within organizations we are beginning to see more and more Communities of Practice, Centers of Excellence, and other formal and informal business analysis groups. Often, individual business analysts within the organization are responsible for leading and coordinating these groups.

The maturity of these groups varies widely. You'll find everything from formally chartered groups—with the executive team investing financial resources in consulting and training to build a BA Center of Excellence—to a passionate business analyst starting a grassroots effort to help fellow professionals share practices via brown bag lunches.

As the groups become more formal, career opportunities emerge in facilitating working sessions, identifying and improving the business analysis process and practices, establishing standards and measurements, improving requirements development delivery, and mentoring other business analysts.

Requirements Management

Another sign of growing maturity in business analysis practices is the use of tools to manage requirements. While the vast majority of business analysts still use word processing and spreadsheet software to document requirements, the use of more advanced tools is growing. Tools are available to visually represent requirements—both in user interface prototypes and in formal models—and to manage textual requirements for delivery and traceability.

Selecting, implementing, and customizing a tool is a significant project at the beginning and involves ongoing work to further customize the toolset as the business analysis practice evolves its processes. A business analyst with a strong process and technical mindset can take on a leadership role in this area.

Strategic Business Analyst

The collection of business analysis activities is often thought of in a project context. As you move up the organization and begin doing preproject analysis, you can open opportunities to work at higher levels within the organization.

A strategic business analyst will probably have a different title—commonly, business architect or management consultant. In these roles, the business analyst helps her organization develop its strategy. Just as a traditional business analyst facilitates stakeholder collaboration to achieve a common vision for a project, the strategic business analyst facilitates executive collaboration to achieve a common vision for the organization. A strategic business analyst might also help the organization identify strategies to execute on specific opportunities, for example, how to integrate a newly acquired company into the business or how to build a new operation to tackle a new market.

"As a manager, you'll have management authority for the individuals on your team and have a responsibility within the organization to ensure that the business analysis team delivers on organizational priorities."

Business Analyst Lead/Manager

As projects become larger and more complex, it's often necessary for more than one business analyst to be involved, and a lead role is required. A business analyst lead defines the project approach; oversees the work of other business analysts, potentially at varying levels of competency; and ensures that the sequence of work and the deliverables produced form a cohesive whole.

Business analyst lead roles can involve a fair amount of coordination, and project management competencies are important. So is the ability to see the big picture, delegate responsibilities, and mentor other business analysts.

Separate from but related to lead roles are managers. As a manager, you'll have management authority for the individuals on your team and have a responsibility within the organization to ensure that the business analysis team delivers on organizational priorities. Where a lead role is often defined within the context of a project, a business analyst manager often will be responsible for several projects at once.

"These five career paths have really just scratched the surface of the possibilities available."

As we find with many senior-level business analyst roles, in years past, business analyst manager simply didn't exist as a stand-alone role. Often, business analysts reported up either to IT managers or, if they were distributed across the organization by business unit, to business operations managers. In some organizations, the project manager is the manager of the business analysts and all the other staff on the project.

In matrix organizations that recognize business analysis as a separate job role and consolidate business analysts in a single department, discrete business analyst manager roles have emerged.

What's Next?

These five career paths have really just scratched the surface of the possibilities available. Just as there is no one answer to "What is a senior business analyst?" there is no one career direction for business analysts. As you build your career plan, consider how you might combine aspects from multiple roles into a position that leverages your strengths and fulfills your passions.

As you begin to look at what's next in your business analyst career, the limits are really only those that you set for yourself. When you think about the competencies that make a strong business analyst, the logical career progression for a business analyst is CEO (an idea I stole from IIBA CEO Kathleen Barret). Whether or not your career aspirations lead you to the corner office, it's definitely true that the skills you learn and the competencies you strengthen as a business analyst create the possibility for a strong future. {end}

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

Virtual Resource Shelf

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

Q: How do you stay connected with virtual team members?

I stay connected with virtual team members by leveraging a globally scalable, unified delivery platform. Application teams can effectively develop requirements that yield predictable applications with higher quality and greater nimbleness in the face of change.

–Filip Szymanski

My favorite way to connect with my remote teammates is in person at our conferences. Between conferences, I like Skype best for the face-to-face interaction it allows, but you can't beat a good, old-fashioned phone call for when you need to get to the point in a hurry.

–Heather Shanholtzer

I use social networks like Facebook and Twitter to catch up on all the casual interactions I might otherwise miss by never bumping into coworkers in the break room, and I use Skype to chat "face-to-face" with my coworkers. Using a few other software tools, I can also share my screen, write notes on my screen like a whiteboard, and draw a virtual mustache and top hat on myself—you know, for illustrative purposes.

–Joey McAllister

I stay connected with other team members with emails and phone calls.

–Lee Copeland

When working with remote team members, a phone with a good-quality headset is my most critical tool. Sure, email, online collaboration tools, instant messaging, and web conferencing are all important, but when you can't meet "face to face" very often, "voice to voice" is the next best thing. A good headset can help you hear better, be heard, and stay comfortable during phone conversations.

–Laura Brandenburg

I think the most transforming factor in enabling good communication with remote team members is video—being able to see each other. There's something about being able to see someone's face—I bet it is something hardwired into us humans. And being able to hear everything going on in the team room when you're remote is so valuable; otherwise, you miss casual but important conversations that people would forget to tell you about.

–Lisa Crispin

George Dinwiddie

Years in Industry: **30**
Company Name: **iDIA Computing**
Email: **gdinwiddie@idiacomputing.com**

Interviewed by: **Don Gray**
Email: **don@donaldgray.com**

Members of a real team know why they're on the team. They share the goals of the team and care about them. They understand who is taking what role, even if the roles are changing fluidly.

I learned that the general flow of XP development was comfortably familiar to me. It's a process of building what you know you want and using that to explore what else you want.

On faux teams, the conflict may be overt and insolvable, but in corporate environments it's more often hidden and unreachable.

> "**The business** of writing tests first was alien to me. How can you test something that didn't exist?"

Real teams tend to get more done than faux teams. That's because they make good use of everyone's talents. Everyone contributes what they can, and everyone helps their teammates contribute more.

It turns out that it's very hard to get someone to do something when they don't see any value in it, but they're quite likely [to do it] when that value is visible from their point of view.

I was less successful at convincing my project lead, so I did "XP For One" for a couple years in an ad-hoc shop. That experience made me pretty good at test-driven development in Java.

> "**The development team** was somewhat neglected by management, so we took it on ourselves to self-organize."

Persistent behavior in groups is an indication that something in the system is rewarding that behavior. This realization gives us tools to make better choices—choices that result in a system that gives the results we want.

I've found that teams that start with an effort to build a "real team" out of a work group tend to succeed much more easily with an agile adoption. Groups that avoid changing the way they relate within the group tend to find little value in their agile attempts.

TECHWELL™

For the full interview visit
www.techwell.com

MAKE YOUR TELE-TEAM WORK

ISTOCKPHOTO

BY LISA CRISPIN AND NANDA LANKALAPALLI

Your company has team members in remote locations. Perhaps people with necessary skills aren't available locally, a key team member moved away from headquarters due to family obligations, or energy costs led management to ask people with long commutes to work from home. For whatever reason, your team now has one or more telecommuters—maybe you are one.

Common wisdom is that having team members in different geographic locations is counterproductive. If you are on an agile team, you might have read that agile only works for collocated teams. Can a team with telecommuters be as successful as a collocated team, or is it doomed to eventual failure?

In this article, we'll look at the physical, cultural, and social obstacles faced by teams with one or more telecommuters—what we call "tele-teams." We'll use real-world examples to explain how you can overcome these obstacles and perhaps even take advantage of the added flexibility that telecommuters can provide.

Our Focus

Distributed teams take many forms. We'll focus on teams where most members work in a main location with one or more people working remotely. This is a situation we have experienced ourselves.

Although the issues may differ for teams in which each team member is remote or teams that have several members in a number of locations, there are a number of variables that affect tele-teams. Let's look at some of the levels of complexity specific to teams with remote members. Then, we'll explain ways you can mitigate the risks and keep the team communicating and collaborating well.

Levels of Complexity

The number of team members working remotely makes a big difference in how tele-teams cope. Fewer telecommuters seems simpler, but a single remote person may be easily forgotten by a large team. The team must ensure that everyone stays in the loop, but your technical solutions may vary according to the number of remote team members.

Time zones are a big factor. Having everyone in one time zone is obviously easier than having team members half a day behind or ahead of the rest of the team. Compromises are required to get some overlap among team members in different time zones. For example, on our team, Nanda adjusts his hours so that he works late into his night, overlapping my morning in Denver.

Roles matter as well. Remote testers, programmers, or DBAs may require different strategies than a remote ScrumMaster, product owner, or product manager. Technical team members have additional communication mechanisms that may either add to or ease their difficulties, such as source code control and continuous integration. Business experts who work remotely might not be familiar with common ways technical teams communicate, such as team wikis.

A remote team member's skill and experience level is an important consideration. If you hire a new programmer who works from a distant location, how will you get that person up to speed on your technology, architecture, and business domain? Since Nanda already knew our application and business inside out before he moved to India, it wasn't a difficult transition for the team. A new hire would have been a different story.

The number of people in the role matters, too. If one of three testers works remotely, the other two testers can provide support and continuity. On the other hand, having the only tester on your team working three time zones away could lead to delays and frustration.

Language is an obvious barrier when a telecommuter's first language differs from that of the main team. Even if everyone is fluent in one language, accents and dialects make a difference. Of course, this can be a problem even with collocated teams, but language differences make remote communication more challenging.

Cultural issues may also impede tele-teams. An informal team on a first-name basis whose members joke around with each other might intimidate a tester used to a formal environment. Behavior that's normal in one part of the world might be unacceptable in another. If you leave work for an hour to go watch your child's school play, will the rest of the team be OK with that? Cultural issues, such as language, can be a factor in any team, but time and distance can increase their impact.

Guidelines

How do we cope with time differences and varied cultural values and social structures when we have telecommuters in various roles with different skill sets? Let's consider some solutions.

Telecommuting policy

Even teams without full-time telecommuters encounter occasions where people work remotely. Traveling team members may work while on the road. Perhaps a bad snowstorm prevents some people from getting to work, someone has to stay home with a sick child, or someone has to wait for a repair person. Situations like these can cause some confusion and floundering—unless you have in place a telecommuting policy with established procedures that everyone can follow.

Chris McMahon published this sample telecommuting policy, based on the experiences of several experienced tele-team members:

> Any team member who wishes will be allowed to telecommute if: The whole team agrees to monitor and be responsive on all of the communication channels required by the organization for telecommuting teams. Team members doing solo or pairing work must keep the rest of the team apprised of their status, and should not be incommunicado for long periods of time.
> Telecommuters who are not responsive on all required communication channels will be questioned and may be subject to having their telecommuting privileges revoked or other disciplinary action [1].

This telecommuting policy also spells out the necessary equipment: "Each team member must have a good quality USB headset and microphone." It establishes the common standard for software needed to communicate with remote team members: "We will use VNC to share desktops." "We will use Skype for audio and Yahoo IM for video." "All team members will monitor the social and project IRC channels during work hours." "We will record all meeting notes and whiteboard photos on the team wiki." The team might experiment with and change technical solutions over time, but everyone knows the current standard.

Team members look to the policy to know when they need to be available and how they should communicate. The team may decide to have "core hours" where everyone on the team, remote or otherwise, is online and in touch by audio and video. The policy might help increase virtual face time with a remote member in a different time zone by specifying meeting times or days when some team members shift their hours. The policy tells business people when and how they can expect to communicate with remote development team members.

Successful implementation of a telecommuting policy requires commitment from every team member. Telecommuting training sessions, conducted in person or via the standard remote communication channels, help establish a workable telecommuting culture.

The telecommuting policy is a living document. Keep it on a team wiki or a similar location where it's easy to access and update. Review the policy periodically to make sure it optimizes communication and reflects the best communication practices in place.

Team discipline

All team members must commit to the telecommuting policy. For example, if the policy says that the standup meeting happens at 9:00 a.m. every day, the base team needs to make sure that all the communication channels are up and running, while the tele-members need to be connected and ready for the meeting at 9:00 a.m.

The biggest problem for a tele-team member is not being in the loop for the decisions that impact him. These could be some simple decisions made during a water cooler conversation or decisions made at a meeting that happens when tele-members are not available. It is important for all team members to think about whether the tele-members need to know about decisions being made. Since Nanda works remotely all the time, the team has written with big letters on the task board “Does Nanda know about it?” This reminds team members that they need to communicate with him continuously.

Keeping development status up to date for all team members is essential. It’s frustrating for the entire team when a tele-member runs out of work or doesn’t know what task to start

Figure 1: Virtual Nanda

next. Whether you use project-tracking software or a physical task board, you need to update it every day. Every day, we post a digital picture of our task board on our wiki. This allows us to keep that “visibility” aspect of agile effective, while providing timely communication with our tele-members.

While these simple tools help us communicate effectively, it’s up to every member of the team to have the discipline to use the tools. It shouldn’t be just one person’s responsibility to make sure that the pictures are posted or an email is sent or whatever communication needs to happen happens. Team members have to avoid a “not my job” attitude and, instead, adopt an “it is my job” attitude with respect to communicating with tele-members.

Though these simple techniques improve communication, they are not the same as face-to-face communication. Our team has set up the “virtual member” virtual telepresence device shown in figure 1 to facilitate face-to-face communication with Nanda. The team assembled a laptop, a high-quality microphone, speakers, and a controllable camera, all mounted on a rolling cart. A tele-member can remotely log on to this laptop using VNC and control the camera. This allows tele-members to be in the team room whenever they feel like it, without the help of the base team. A remote person can participate in meetings and be rolled over to individuals for pairing or conversations. The microphone allows the tele-member to hear casual background conversations that he might want to join. This encourages the tele-members to be more in the loop.

Larger teams require even more focus to keep remote members in the loop. It’s too easy to forget the tele-members. If your team has twenty people and only one is remote, consider dividing into two or more teams.

Who can telecommute?

As we discussed earlier, role and skill level impact the success of a tele-team. For example, a database administrator or system administrator can usually work independently and may find telecommuting easier than programmers or testers who collaborate constantly.

If your team has a remote product manager or product owner, you’ll have to do extra groundwork to finalize requirements for a user story or feature before development starts. In this situation, changing requirements in mid-development is more difficult and disruptive. You have to make accommodations appropriate to the roles of the telecommuters on your team.

No matter what your role, if you’re working remotely part or full time, you need to know how to set up your own computer, network, and any other components. If your hardware or software fails, you should be able to bring everything up and running as quickly as possible. Having redundant hardware can help minimize downtime. A telecommuter also needs to know how to connect to various servers at the base datacenter—or at least know where to get the information.

A generalist is usually better suited to telecommuting, but suppose your team hires a hard-to-find specialist who can’t move to your location. We recommend that you bring the new hire in for on-site training with the team for as long as possible before she starts telecommuting. More face time is always better.

We’ve already mentioned techniques such as posting photos of the task board on a wiki and setting up a “virtual team member” telepresence device. In addition to technical solutions, it’s a good idea for a tele-developer to have a buddy on the base team who facilitates communication. This could be another programmer or tester with whom he is pairing on the story.

Missing out on hallway discussions where decisions are made disrupts tele-members regardless of their role. If a developer and product owner decide to change a piece of functionality and forget to tell the tester, she may waste time testing to the wrong requirements, possibly reporting unexpected behavior as a problem when it is actually changed functionality. The high-quality microphone we mentioned earlier helps remote people get in on these conversations. Consider putting a webcam and microphone in the break room, as well.

Design meetings in front of the whiteboard are a great way to communicate, but how can tele-developers see the whiteboard real-time? Teams can use smart whiteboards that transmit to a remote whiteboard or use web-based whiteboard applications. Some webcams have a high enough resolution to transmit the whiteboard drawings.

Our team uses a cheaper solution: One team member types the important points during the meeting over a "backchannel" chat that doesn't disrupt the meeting or updates a wiki page that Nanda and anyone else working remotely that day can see. At the end of the meeting, digital pictures of the team are posted.

Challenges

Since development teams usually have more programmers and testers than any other role, let's look at specific issues for telecommuters in these roles.

Specific challenges for programmers

A remote programmer needs to communicate a lot with the base team to stay on the same page regarding the requirements and day-to-day progress. If you're a programmer working in a different location, you need to understand all the work planned for the next several days to avoid any idle time when you have finished the task in hand. If you do find you've run out of coding tasks, be willing to take on testing tasks or other activities outside your normal job so you can keep the team moving forward.

Tele-programmers need to understand the other programmers' viewpoints of the functionality on which they're working. Have face-to-face video conversations to get consensus architecture and design decisions. This is one area where a virtual telepresence device comes in handy. Pairing remotely helps keep everyone engaged and is easy to do with desktop sharing devices and VOIP or phone.

Frequent check ins are a good practice for all developers, but it's especially critical for remote members to check in their work and keep their source code and local build environment up to date. It's helpful if all team members use the same IDE so it's easier to help the remote members when they have problems.

Specific challenges for testers

One of the most common problems for any team, whether or not it has remote testers, is for testing to be squeezed to the end of the iteration or release because coding took longer than expected. Having testers in a different time zone may aggravate the problem: The tester finds a "test-stopper" issue while the rest of the team is asleep and has to wait an entire day for a fix. This is one reason why testing can't be a "phase" but must be integrated throughout development. It's also an argument against having the entire test team working in a location different from the programmers.

By using a visual task board to focus on finishing one story at a time, testing activities can be spread throughout the iteration rather than concentrated in the last couple of days. The team needs discipline to limit the amount of work in progress. Remember, no story or feature is "done" until it's tested.

Testers often don't have the same technical skills as programmers, so make sure someone's available to help a remote tester who gets stuck on a continuous integration (CI) or test environment problem. On a team where Lisa worked remotely, she ran into situations where the CI process broke down and she couldn't build the software on her own environment, either. She couldn't deploy and test the latest build until someone on the base team could help address the issue. Invest in a rock-solid build-and-deploy process and stable test environments.

If you're a telecommuting tester, grow your technical skills and domain knowledge so you can be more independent. Don't be afraid to ask for help when you need it so you don't slow down the team.

Embracing Good Practices

Development practices popularized by the agile movement are especially beneficial for noncollocated teams. CI is a must for any team. A good CI build-and-deploy process keeps everyone productive.

Upgrades to the source code control, CI software, or third-party libraries must be well coordinated with remote team members so their progress isn't impeded by difficulties building or deploying the application. Our team maintains up-to-date documentation on build-and-deploy procedures, third-party libraries, and source code control information on the team wiki, alerting teammates to changes by email and in the daily standup. Our continual integration process includes sending emails to all team members with results of each build.

As mentioned in the previous section, pairing is just as effective when done via desktop sharing, audio, and video software. Pairing goes a long way toward helping tele-members stay in the loop and learn domain and technical knowledge. The same communication technologies make test-driven development at both the unit and acceptance level work well for tele-teams. These practices help overcome the disadvantages of having remote team members.

Daily standup meetings help everyone on the team stay on the same page, as do frequent brainstorming and planning meetings. Timebox the meetings, especially if a remote member is getting up early or staying up late to join in. Schedule retrospectives—critical to team success—at times when tele-members can join in.

Top Tips for Tele-Teams

Telecommuting may allow your team members a better work-life balance, help your business retain a key team member who must move away for family reasons, or let you take advantage of top talent outside your local region. The technology and practices that help tele-teams communicate are mainly good communication tools and techniques that would help any team. Our list includes: Set up a virtual member telepresence device so that remote people can effectively work in the team room with the base team; establish standard tools for desktop sharing, instant messaging, and other voice, video, and back-channel communication; and consider tools for the occasions when team members must work from home or while traveling.

Our team has found that the simplest solutions usually work best. Using a physical task board and whiteboards for planning and understanding requirements and posting high-resolution photos of these on the team wiki in a timely manner keep our tele-members up to speed. We like to include snapshots of daily goings-on so Nanda can see what the team is up to. For example, Lori converted her cube into a standup desk, Mike got a new haircut, and Vince's wife made a birthday cake. Little touches help every team member feel equally valued, regardless of location.

Some teams may find that online planning and tracking software works better for them. There are many open source and commercial solutions available. Don't hesitate to experiment and find what works best in your particular situation.

We can't overemphasize the importance of "face time." It's a good idea to bring all the tele-members to the base team periodically to help them get in sync with the rest of the team. Take advantage of any opportunities for a base team member to visit the tele-team member. Lisa had the chance to go to India to present at conferences and was able to visit Nanda and see where he lives and works. It helped her better understand his challenges working half a world away from the rest of the team.

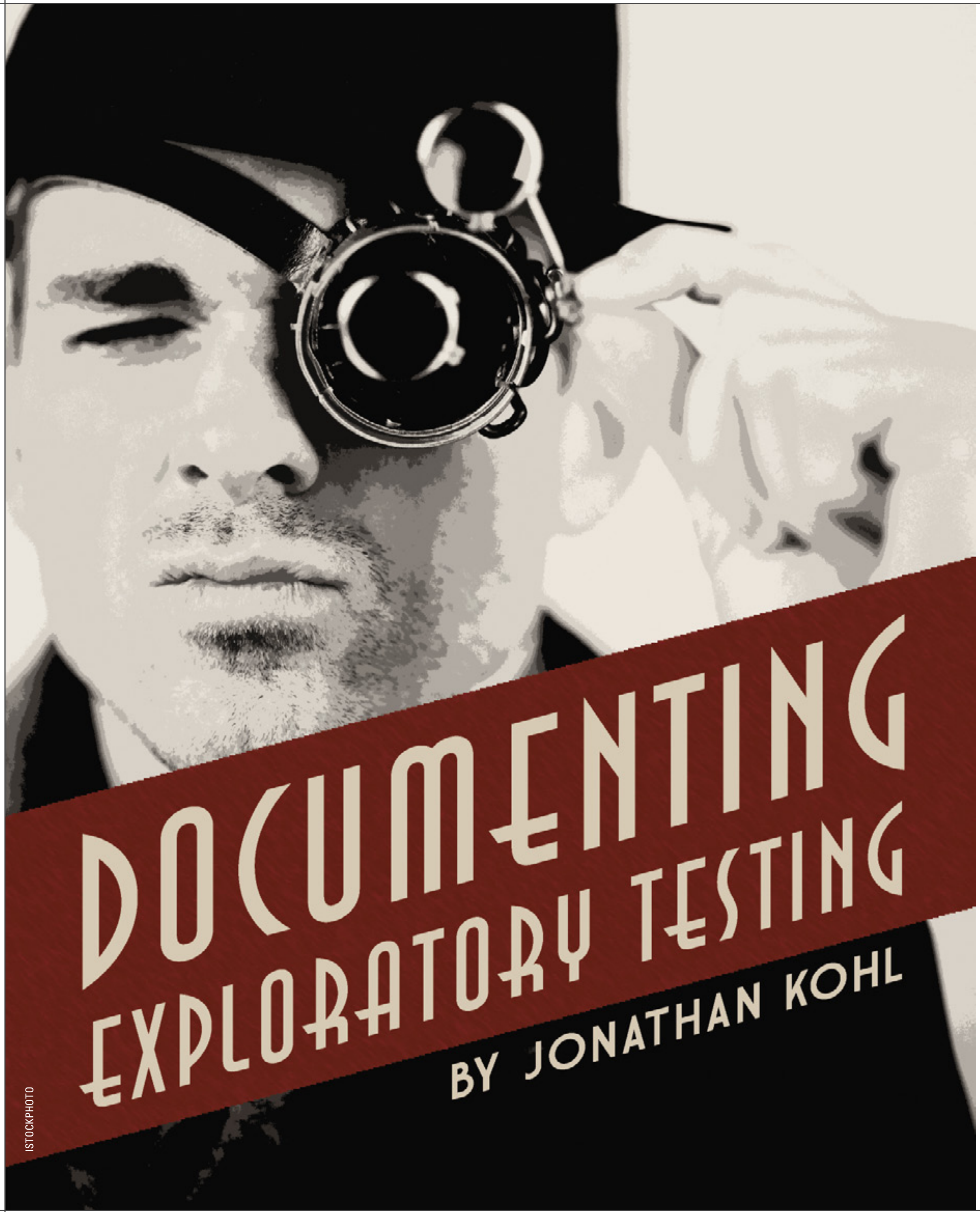
Making your tele-team work will require patience, discipline, and commitment. But it's fun and rewarding. Every iteration—or even every day—take time to evaluate how well everyone is communicating. What's getting in the way of the remote team members? Is it hard to keep everyone up to date? Focus on one problem at a time and experiment with a solution. If that experiment doesn't work, try something else. Tele-teams often enjoy more diversity, which leads to more creativity and innovation. Better communication is good not only for the remote people but for the whole team. You may find that your tele-team provides more value to your business than a traditional collocated team could. **{end}**

lisa.crispin@gmail.com
nanda.lankalapalli@eplanservices.

Sticky Notes

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

- References



DOCUMENTING EXPLORATORY TESTING

BY JONATHAN KOHL

JUST BECAUSE WE ARE USING AN EXPLORATORY TESTING APPROACH ON OUR PROJECTS DOESN'T MEAN THAT WE DON'T DOCUMENT OUR WORK. With exploratory testing, you can document as much or as little as required by your stakeholders.

Determining documentation requirements on software projects isn't difficult; it just takes a bit of investigation. For example, while working on a project in a regulated environment, I talked to the people in charge of regulatory issues for the company and asked them what documentation they needed. They said they needed a risk assessment, a test plan, test cases, and test results. I had everything but test cases, so I pressed further. They wanted test cases simply because that's what they were used to. The regulatory requirements didn't ask specifically for test case documents; there just needed to be enough documentation so that tests could be repeated by others.

Planning and Strategy

WHAT DOES A TEST PLAN LOOK LIKE IN EXPLORATORY TESTING?

Often, test plans that direct exploratory testing look very similar to their scripted testing counterparts. On any project I am leading, I go through analysis and planning to determine how to optimize the use of our people, tools, equipment, budgets, etc. A risk assessment (required by auditors) can help focus our testing.

To create a risk assessment, I do some research. I talk to marketing, sales, and product management and ask them about core functions of the product. What is the purpose of the software? What kinds of tasks would users expect to perform with the software? What would product success or failure look like?

Next, I talk to the technical team. What features are new? Are there any technology changes that are required, such as new tools, libraries, or hardware? Have they discovered any challenging areas in their work? How do they define success and failure?

I then talk to product management about quality criteria. What characteristics are they expecting from the software? What about technical areas, such as security, accessibility, and performance?

The risk assessment can be a list of items describing how we would mitigate those risks through testing, a focused testing approach for a specific risk, or a recommended technique, such as performance or security testing. Figure 1 shows an example of a risk assessment.

After developing the risk assessment, I look at a test strategy. What is the purpose of our testing? Sometimes, project stakeholders have a clear objective for testing; other times, it isn't as clear. I provide options to stakeholders. "Do you want us to find important bugs as quickly as possible, or are you more concerned that our testing mitigates the risks outlined in the risk assessment?" Once I have determined a purpose for our testing work, I can look at who is available and start to outline what tools, practices, and tactics we will use when testing.

Next, I look at project resources. Which people are available and when? What tools, machines, software, and other resources do I need to complete this project? Once I have clear answers to these questions, I start to glue together my project analysis, risk assessment, strategy thoughts, and logistics. I learned this from James Bach [1]. If this

Session Tester 0.3 Release Risk Issue	Potential Consequence or Outcome	Outcome Tolerable?	Probability of Occurrence	Mitigation Approach by Testing
Business Related				
New workflow: application loads in full screen mode, not session-start mode.	Usability may be impacted negatively. The application may load slowly.	No	Medium	*Scenario testing on all supported platforms, focused exploratory testing*
Text editor changes	The notes that are entered may not be saved accurately. There may be a performance impact since we are using a 3rd party library instead of a lightweight parser.	No	High	*Regression test simple sessions. Focused ET using test data generation to try different types of data, different sizes of data, and each markup tag,*
Project workflow is new	The application may not work properly. It might stop responding, crash or behave strangely. Users may not understand the workflow.	No	Low	*Functional Testing, User Scenario Testing, Usability testing. Focused ET in error generation workflows.*
View/edit sessions is new	Time-sensitive information may get corrupted in edit mode.	No	High	"
Technology Related				
New XQuery library for reporting	Reporting information might get lost or corrupted.	No	Medium	*Testing using generated XML session docs using different charactersets, time-related data.*
New SDK used for development	Some platforms (notably Mac) may not support all the features.	No	Medium	*Functional and Regression testing on each supported platform*
Using syntax library as text editor	We may use the API improperly, causing unintended consequences for end users.	No	Medium	*Requirements-based testing, user scenario testing, data generation using a tool.*

Figure 1: Sample risk assessment

needs to be formalized in a document, this becomes the basis for my test plan.

Traditional testers love test plans, but many exploratory testers hate them. I have a secret: I need to do all of that work, anyway, so if I document bits of it as I go, taking that existing information and plopping it in a test plan document is simple and fast. If I need to change the test plan into a different format or adapt because the project changes mid-course, it's easy and quick.

Guidance Documents

How do new people learn without test cases?

Test cases aren't necessarily the best way to learn how to test new software. Alternatively, I use tools and techniques for learning that are proven in training and documentation disciplines. For example, in the regulated-environment project discussed earlier, instead of relying on test cases, we created a "guidance" folder on a shared network drive. I found existing sources of documentation from the marketing team (sales brochures, demo presentation materials), the documentation team (in-progress user manuals), and the operations team (installation and administration guides). That gave us a start but didn't provide enough information for testing. Using documentation department templates, we started writing "how to" documentation with a testing focus. We used screen-recording software to record demo videos since they were easier to maintain than written documentation. If specific setup was needed, we created "calibration" documents that provided step-by-step instructions. To help with test idea generation, we created testing cheat sheets with different testing ideas.

To help focus testing in specific areas, guidance can be provided in the form of checklists. Each coverage model can generate checklists with corresponding guidance information in written and video form.

Checklists essentially are lists of test ideas. The lists are specific enough to provide focus but vague enough to allow for personal interpretation and freedom to explore. Figure 2 shows a sample checklist.

This is how a tester would use documentation to help guide his exploratory testing: He would look up a specific test checklist related to the area of the program he was testing. If he was unfamiliar with that application or feature set, he would read a how-to guide document to learn how to use the software and watch videos that demonstrate how to use and test the software. Next, he would start testing the items listed in the checklist, referring to the guidance information as needed.

When he needed more test ideas, he'd review testing cheat sheets to help trigger more creative ideas.

Documenting Test Execution

How can we demonstrate what was tested?

Session-based test management is a high-accountability approach to exploratory testing invented by James and Jon Bach. This is a descriptive approach to testing, where you document what you are testing as you go. The traditional test case model is prescriptive—you document in advance what you would like tested. Both approaches generate documentation that can be reviewed and audited. I use lightweight implementations of session-based test management. Session ideas focus on specific testing tasks. Testing sessions are uninterrupted periods of testing time lasting from sixty to 120 minutes. After an exploratory session, testers review their session findings with colleagues. This helps generate follow-on test ideas and is great for knowledge transfer. If there are regulatory or auditing requirements, a session sheet can be used as a project artifact.

If traceability is needed, session sheets can map to coverage outlines and to guidance documents and media. For example, an auditor could ask about a specific bug that was logged and be referred to the session where the bug was discovered. From the session sheet, the auditor could review the coverage outline checklist and the guidance documentation. Watching a short video of the bug report and contrasting it with the how-to testing document and video, he could repeat the test himself and verify that the bug had been fixed. See figure 3 for an example.

While the auditors on our example project were at first a bit concerned with a lack of test cases, they loved the use of media and the "how-to" document approach. They found it easier not only to get a sense of what was tested but also to repeat tests themselves. The video was incredibly useful for them, and they liked that session sheets described what actually happened instead of test cases that describe what we expected to happen.

Documenting testing using video has become more popular. It's an inexpensive and powerful way to show people what you mean rather than trying to describe concepts in a document. Recording technology is cheap and readily available. There are also free software options on a variety of platforms. One of my colleagues recorded, produced, and distributed a testing training video of me demonstrating test techniques—all on his smart phone.

Session Tester 0.3 Coverage Outline

Revision:	r242			
Application Load				
Full application defaults	pass			
Treeview	pass			
Session buttons other than "Start" disabled	pass			
Project				
New Project wizard				
Project Tree View actions				
Edit Project	pass			
New project with tags				
New Session				
Session buttons enabled	pass			
Timer	pass			
Date/Time	pass			
Mission	pass			
Primer	pass			
Project relationship	fail			
Session State				
Text Editor				
Timer	pass			
Autosave				
Tags				
Pause				
Stop				
View session				
Reports				
Html Reports				
Structure	fail			

Figure 2: Sample test checklist excerpt

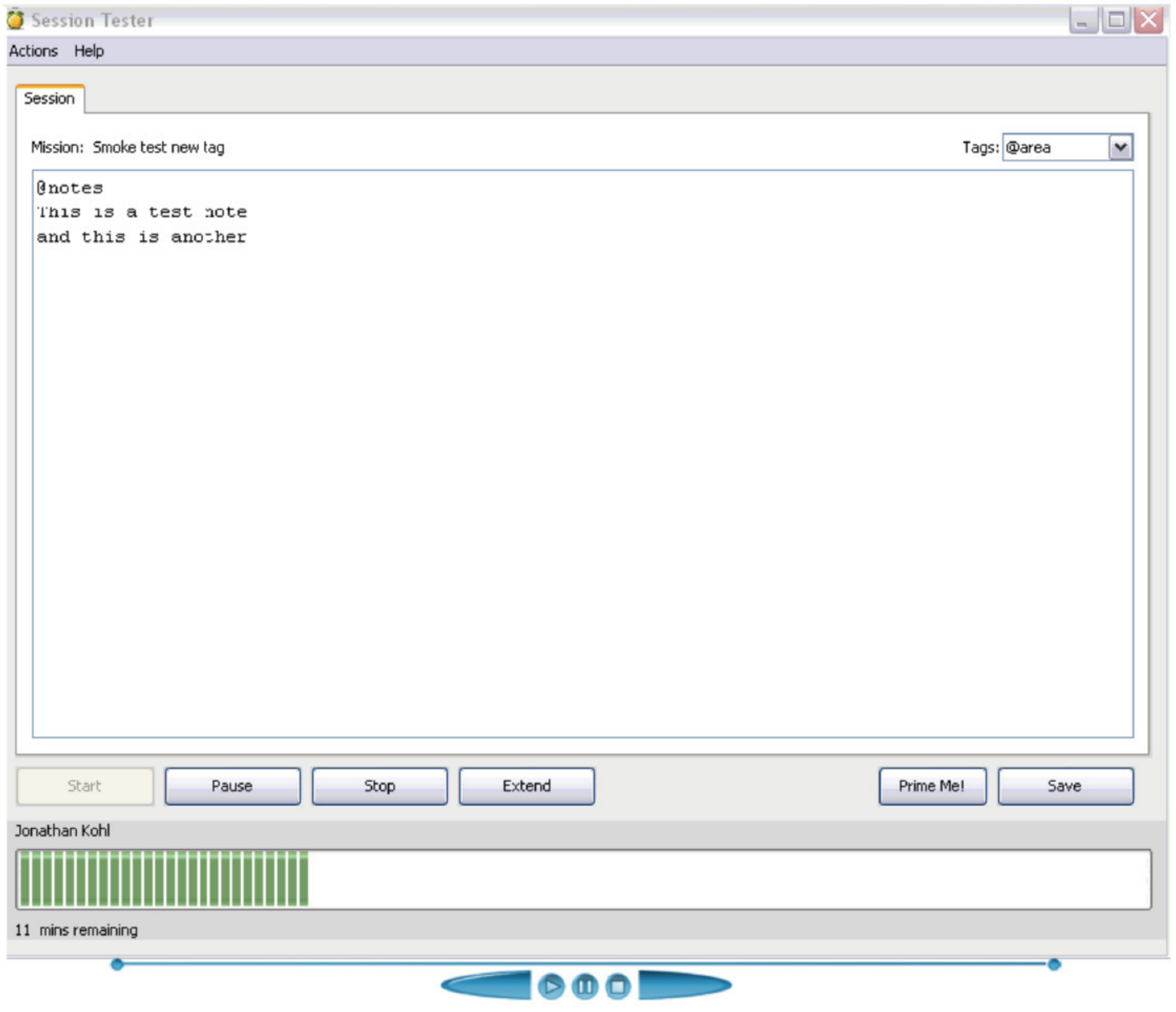


Figure 3: A test video screen capture

Documenting Results

HOW DO WE TRACK RESULTS AND REPORT ON TEST COVERAGE?

On a project that uses free-form exploratory testing without documentation, it can be difficult to explain what was tested. There are lightweight tools that can be used on exploratory testing projects to demonstrate results and progress.

I like to have visible results that show quality and test coverage in simple terms. From there, more detail can be requested and easily supplied without providing much burden on the team. James Bach has a "low-tech" testing dashboard template that can easily be transcribed on a whiteboard. Any team member walking past the whiteboard can get a sense of current quality and testing coverage on the project. From there, checklists can show more detail and session sheets can show actual testing results. This process can easily fit with project management, fault tracking, and other team productivity systems.

I prefer a "pay-as-you-go" approach to documentation. I don't do a lot of speculative, up-front documentation if I don't have to. I want to avoid having to change it frequently as the project adapts and changes. I do a bit of documentation at a time and build toward final products as the project itself evolves. If you are using an agile or iterative lifecycle, you may not need to document all the time, particularly if you start testing at the beginning. If you are in a regulated environment, auditors are used to a "testing phase," so you may be able to use an iteration near the end of the project to create required documentation.

Conclusion

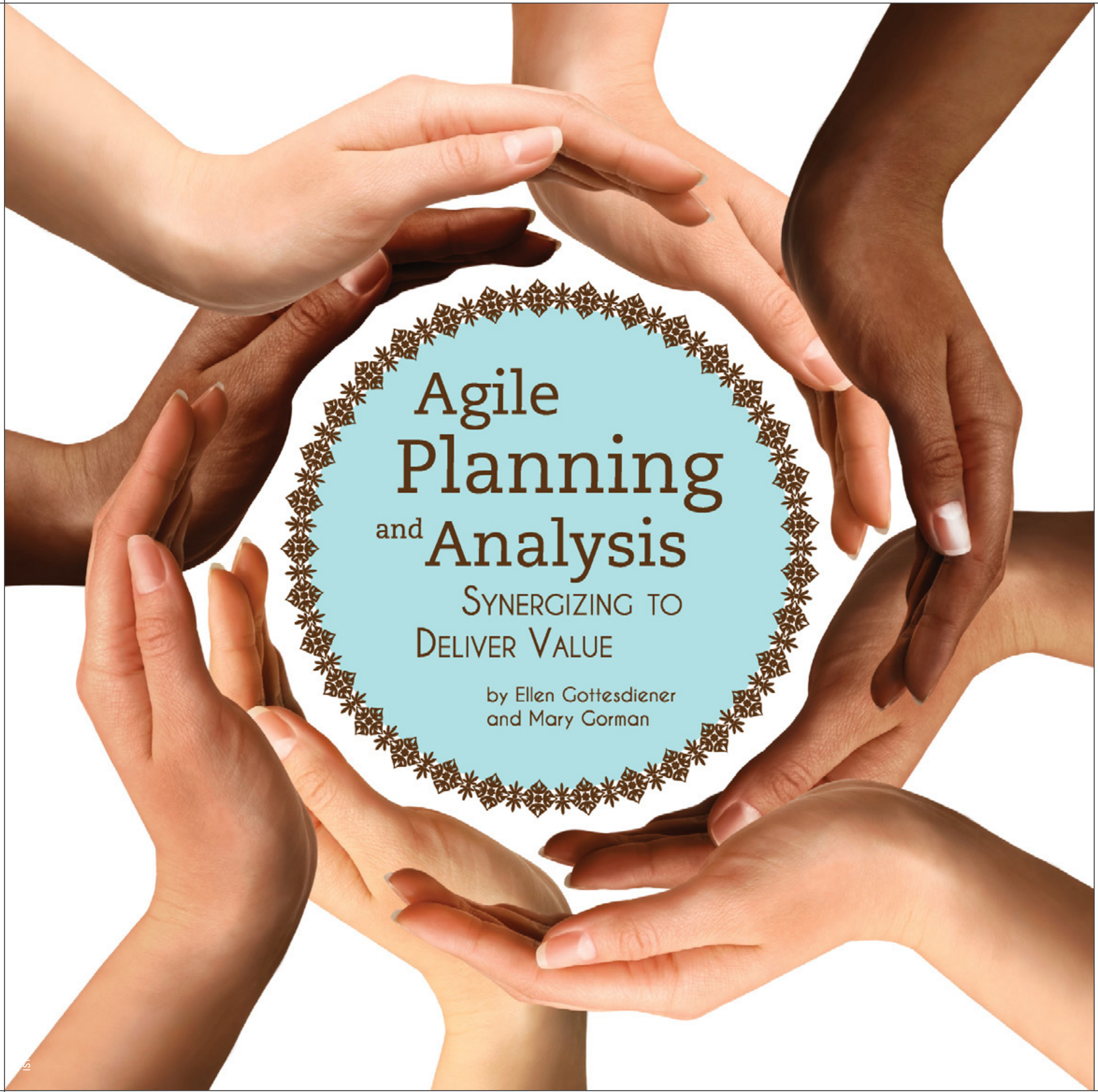
You can use as much or as little testing documentation as you need on an exploratory testing project. However, make sure that what you do is compatible with the lightweight, test-execution focus of exploratory testing. Do not impose documentation overhead that dominates testing activities—those should focus on testing first. In many cases, you can use something lightweight to solve a problem that seems to require a heavyweight approach. With a little creativity, you can fit your documentation requirements within an exploratory testing approach. **{end}**

jonathan@kohl.ca

Sticky Notes

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

- References
- Further reading



Agile is about the continuous incremental delivery of valuable, market-ready software. Your agile team iteratively explores and evaluates product needs—commonly referred to as requirements—by planning and analyzing what to build, defining acceptance criteria, and then building and testing product increments. A crucial aspect of your work is planning—and planning to plan—while integrating just-enough, just-in-time analysis.

Analysis and planning are synergistic. They are coordinated efforts, and one feeds the other. Analyzing requirements deepens your understanding of product needs so that you can identify and select the most valuable ones. Planning is the allocation of those product needs into delivery cycles, given your limited capacity (people, time, money, resources). Together, planning and analysis seek to maximize business value.

The Backlog: The Basis for Planning and Analysis

The *backlog* is a master catalog containing a prioritized list of unfulfilled product needs at varying levels of granularity. Figure 1 shows one way of categorizing backlog items.

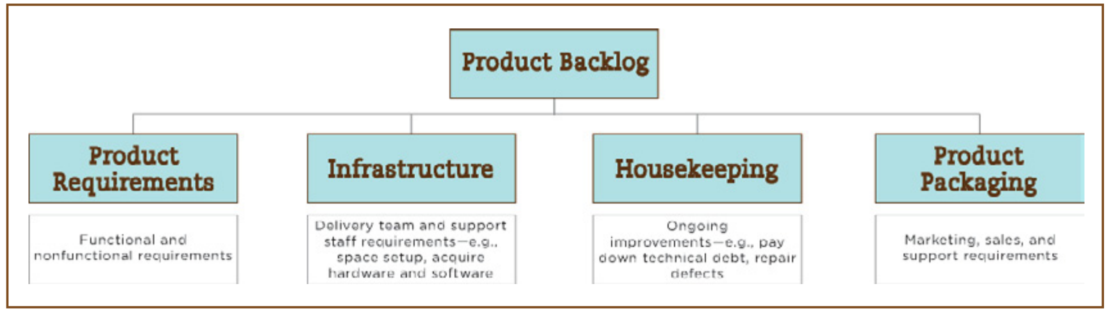


Figure 1: Product backlog categories

Typically, most of your backlog items will be product requirements in various formats: user stories, one-line titles or story descriptions, drawings or sketches, and so on. Items in a healthy agile backlog are what Roman Pichler calls DEEP: detailed appropriately, estimated, emergent, and prioritized [1]. Note that "detailed appropriately" means that, at any given moment, some items will be highly detailed and others less so.

The backlog is dynamic. Items are added, removed, altered, reprioritized, deferred, decomposed, or prepared as needed. This ongoing planning and analysis of

Internal Stakeholders
• Sponsor
• Product champion
• Business, project, and product managers
• End-users
• Indirect users—e.g., performance improvement staff, down/upstream interfacing systems staff and management
• Delivery team members—e.g., architects, data and database administrators, business analysts, developers, testers, user experience experts, etc.
• Support staff—e.g., help desk, installers, operational staff, maintenance staff, trainers, network operations, etc.
• Advisers—e.g., compliance, HR, legal, subject matter experts
• Sales and marketing staff
External Stakeholders
• Buyers (purchasers)
• Direct end-users
• Regulatory agencies
• Competitors
• Third-party suppliers and partners
• Shareholders and investors
• General public
• Media

Figure 2: Stakeholders who offer perspectives on the product backlog

View	Description
Big-view	• Provides a holistic understanding of product needs you may deliver over a long time horizon to satisfy the product vision and business goals
Pre-view	• Defines product needs for the near term
Now-view	• Defines immediate product needs in sufficient detail to estimate, build, and test what will be delivered next

Figure 3: The three views of product needs

backlog items is known as *grooming*, *pruning*, or *refining* the backlog. As you do this work, you must be vigilant to ensure that the backlog items align with the product's vision and business goals, realizing that goals may change over time as the organization, market, and competitors evolve and you get feedback from users.

Grooming maintains a runway of product needs that are ready to pull into planning for the next and future delivery cycles. The trick is to balance current and future planning. We find that teams typically work two to four iterations ahead; the further ahead you're planning, the less detailed the requirements will be.

Whichever agile or lean framework, method, or technique you use to analyze the backlog, you may also employ *artifacts*, such as personas, a data model, a story map, or business rules. These artifacts, kept as lightweight as possible, can be very helpful as the team explores, designs, builds, and tests a slice of the product.

Power of Perspectives

Many technical and business people think of requirements as specifications that get defined and then "thrown over the wall" to the technical people. But on agile teams, this classic view is altered. Product needs are explored and evaluated through a partnership of technical and business people so that team members can collaboratively understand and deliver business value.

It takes a shared understanding to plan and analyze product needs, including perspectives from cross-functional disciplines. The team needs to incorporate the input of the internal and external stakeholders listed in figure 2.

To lead the exploration and analysis, many agile teams rely on a few people who have strong analysis and domain skills. In our experience, people with these skills include business analysts, product managers, testers, user experience experts, and the like.

The Three Views of Product Needs: Plan and Analyze the Backlog

As you pull and evaluate items from your backlog, the key concept is that the level of detail of any item will vary depending on the amount of lead time in your planning. The closer you are to building a product need, the more detailed it should be. You can't know all the details of all the backlog items up front, so you sketch out the long view of the product to establish a common focus and marshal organizational resources (people, money, space, governance). Then, you iteratively define what to build now and what to build next.

Product champions and development teams tend to think of and refer to the product backlog from three points of view based on where a given product need is within the timeline of the development cycle, as shown in figure 3. We call these three views the *big-view*, the *pre-view*, and the *now-view*.

Planning and analysis get increasingly fine-grained as you descend the view hierarchy. The big-view idea gets more refined at the pre-view level. Then, in the now-view, it is sufficiently detailed so that the delivery team can estimate the next delivery cycle, develop acceptance tests, and design the solution.

The Big-View

A product need starts at the big-view as a general idea—for example, a feature that you think will fulfill some aspect of the product vision.

Note that the big-view reflects how the new product will fit with the other products in your organization's portfolio. Thus, the big-view should align with your organizational strategy and should be feasible.

The Pre-View

The pre-view defines product needs in enough detail to support planning for the next release. The pre-view is informed by the big-view and the product roadmap (an artifact that describes your planning decisions). The pre-view for delivery teams with short release cycles can be as near term as a day or week. For other teams, it is one to several months. Each release contains chunks of consumable, marketable, valuable features.

The Now-View

The now-view describes product needs in sufficient detail that your team can make reasonable estimates of the work needed to deliver those requirements to a predefined set of acceptance criteria (conditions of satisfaction). The now-view is informed by your release plan and your most recent retrospective. Once you've planned your now-view and are in the midst of delivery, the now-view entails analyzing product needs under development sufficiently to design, test, and deliver the product.

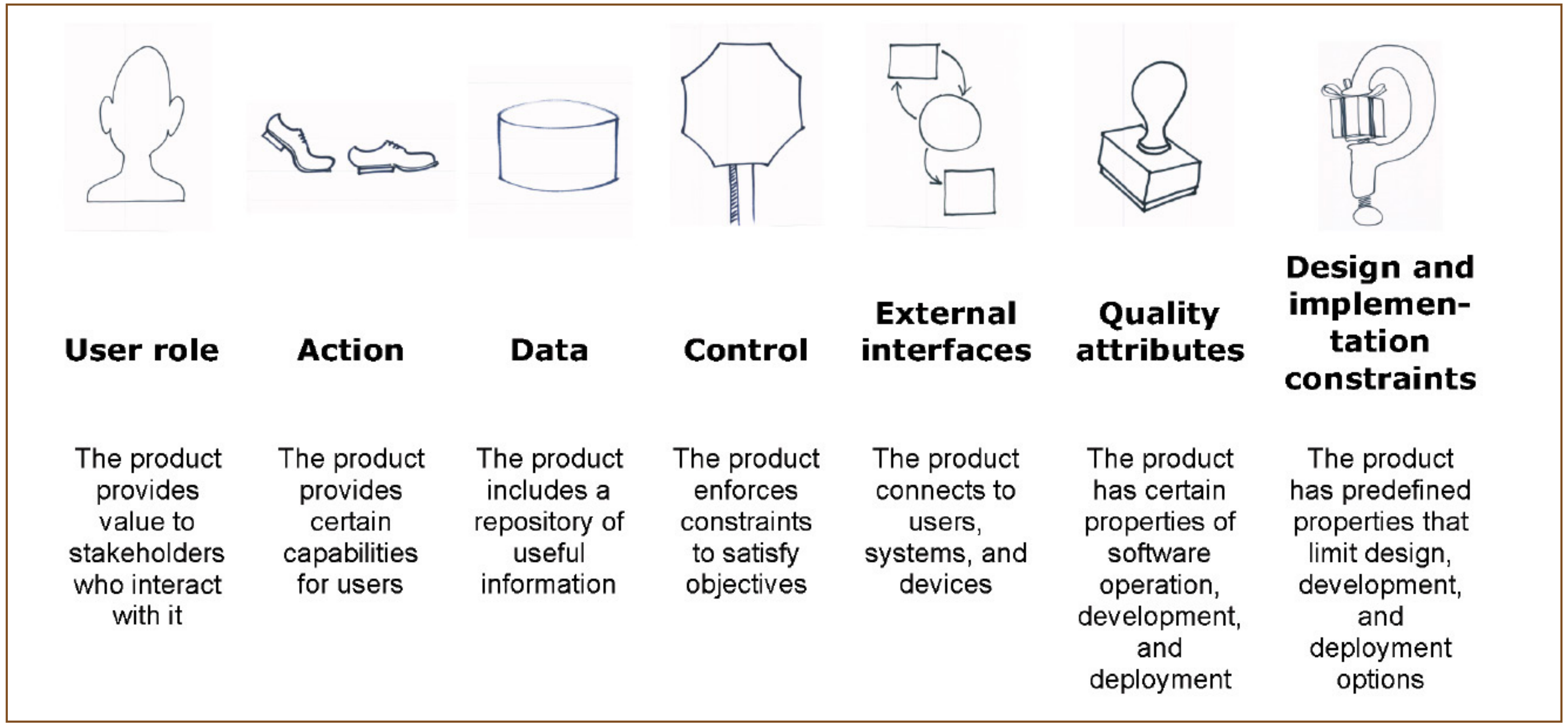


Figure 4: The seven product dimensions

Analyzing the Backlog Across Seven Product Dimensions

In partnership with the product champion and other stakeholders, the team works together to analyze the big-view and pre-view backlog items in just enough detail to estimate and plan. For now-view planning, the delivery team analyzes and estimates the activities needed to build, test, deliver, and validate the high-value product needs.

To assist in this analysis, we've developed the seven product dimensions shown in figure 4. They serve as a checklist that technical and business partners can use to make sure they're analyzing all the key aspects of a product backlog item. The seven product dimensions help you structure daily conversations about backlog items and give you a comprehensive understanding to guide planning. You can use this construct to analyze work in all three views (big-view, pre-view, and now-view).

The team looks at a backlog item, asking—in any order—Who is the user? What actions does the user perform with this item? What data goes in and out? What controls does this item enforce? What users, systems, and devices does it interface with? What quality attributes must be satisfied? What are the design and implementation constraints that apply?

There are a variety of ways to structure this conversation. We like to use an explore-and-evaluate technique in facilitated workshops; the product partners collaboratively identify timely product options (explore), assess the options and identify the highest-valued ones (evaluate), and slice each product need based on value.

In this way, the team and product champion explore and evaluate options for each product need at the level of detail appropriate to the view. This approach engages you in creatively using right-brain thinking—visualizing options, relationships, dependencies, and flow. You might start by working on the wall, creating lightweight, organic analysis models such as a data model, state diagrams, a context diagram, a dependency map, decision tables, prototypes, and so on. Documentation, too, can be lightweight—posters, photos, wikis, and the like—to serve collective memory. These artifacts may be temporary or, if valued by the partners, may become the basis for building and packaging the product.

Throughout the conversations, the entire team gains crucial knowledge. As the product champion assigns a value to each product need, he asks the team to gauge the effort it will take and the risks of implementing it, improving his understanding of technical concerns and the development process. As team members discuss the product needs, they learn more about the business context for the work. Team members are responsible for questioning, challenging, and clarifying the product champion's filtering criteria, deepening their understanding of the business domain and the options that will provide value.

Together, team members balance their understanding of options for each product dimension of each backlog item with a hard-nosed assessment of which options are most valuable for the next delivery cycle. Ongoing analysis grooms the backlog and shapes the plan for each view.

Successful Synergies

Agile planning and analysis are interdependent and synergistic. Working hand in glove, they provide stakeholders with a flexible structure for continuous delivery of value. **{end}**

ellen@ebgconsulting.com
mary@ebgconsulting.com

Sticky Notes

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

- References

Ranorex 3.0

GRAZ, AUSTRIA—Ranorex 3.0 has been released and is now available for download. This new version features a test suite view that allows easy test case management and simplifies the combination of recordings and code modules within tests. Ranorex Studio now supports data-driven testing via connectors to SQL, CSV, Excel, and simple internal data tables.

The test suite view simplifies the combination of simple modules and recordings within single test cases. In addition, one can directly connect test data—stored in tables or external data sources—with variables used as place holders within modules and Ranorex repositories.

One important goal of the new version is to reduce effort in maintaining test automation projects. The improved Ranorex repository browser now provides new functionality, allowing you to quickly search for and filter items and to clean up the repository in case of unused elements. In addition, the use of variables within RanoreXPath simplifies test automation of language dependent user interfaces.

ranorex.com/products/whats-new-in-ranorex-3.html

eggPlant Mobile Testing Solution

BARCELONA, SPAIN—TestPlant, in partnership with Real-VNC, has launched the eggPlant mobile testing solution, which extends eggPlant's reach into software running on smartphones and tablets across the major mobile operating systems.

eggPlant uses RealVNC's VNC Mobile Solution to connect directly to a device. A true user experience test is created, as eggPlant is capable of recognizing icons, colors, text, and fields on any display. A test script is automatically generated, as eggPlant is trained to move around an app using its image store to spot and compare display prompts. Multiple tests become effortless, and eggPlant reports test failures with helpful screen images of the exact point of failure. An automated test script can be created on one device (or emulator) to be repeated many times on other devices.

testplant.com

CollabNet TeamForge Project

BRISBANE, CA—CollabNet has announced the general availability of CollabNet TeamForge Project, an integrated suite of web-based development and collaboration tools for agile software development. The general availability release follows a four-month beta test by more than one-hundred organizations. TeamForge Project is a component of CollabNet's Codesion Professional Edition product, a secure, instant-on developer cloud that enables developers to instantly provision Subversion, Git, and other code repositories, as well as bug tracking, agile management, and full application lifecycle management capabilities.

www.open.collab.net/community

Sitefinity 4.1

WALTHAM, MA—Telerik has revealed details of its latest release for Sitefinity, the company's robust web content management system. The Sitefinity 4.1 release introduces a number of upgrades and improvements, including enhancements delivering performance increases of up to 35 percent. Also being debuted is support for Windows Azure, Microsoft's cloud services operating system, and native language packs for international Sitefinity users.

Enhancements incorporated into the Sitefinity 4.1 release include a new intelligent auto-caching algorithm enabling faster page loading and serving for websites; an upgraded virtual path provider, which streamlines and accelerates serving of dynamic content; and smarter auto-caching and an improved virtual path provider.

New language packs, providing authors, content managers, and content creators with a broad variety of native language support for backend and administrative tasks, are also being launched. Other features, upgrades, and enhancements in this release are:

- Comprehensive lists module
- Customizable user profiles for storing and displaying user-specific information
- Support for page redirects and multiple URLs
- Shared content blocks permitting greater content reusability
- Improved ORM mapping for greater extensibility

sitefinity.com

TestRail 2.3

Gurock Software has announced the release of TestRail 2.3. TestRail helps software development teams and QA departments manage, track, and organize their software testing efforts. TestRail 2.3 comes with major scalability and performance improvements for large teams and introduces new optimized views for test suites and test runs, and many other improvements.

With more than sixty changes, improvements, and bug fixes, TestRail 2.3 is the biggest update of the product since the release of TestRail 2.0. The new version comes with alternative views for the test suite and run pages that make it easier to work with larger numbers of test cases. An optimized user interface with a new section tree simplifies navigating and organizing test suites and rearranging test cases and sections via drag and drop.

The new version comes with improved support for third-party tools and Internet Explorer 9, and a new defect plugin for Axosoft OnTime to directly push and look up issues and bug reports. Other improvements include optimized Excel exports, easier attachment backups, and a new way to customize the subjects of email notifications.

gurock.com/testrail

GUI*dancer* 5.0

BREDEX GmbH has announced that version 5.0 of GUI*dancer* has been released and is available for immediate download.

GUI*dancer* 5.0 contains the new Test Style feature to offer advice on structuring and naming tests. The Test Style component consists of the best practices taught to customers during seminars, workshops, and coaching sessions to ensure long-term readability and maintainability of test projects. GUI*dancer* 5.0 users will continue to have access to the BIRT reporting function for long-term quality analysis, the Mylyn integration for context-based test specification, and model-driven testing to import test case structures from UML diagrams.

bredex.de

Gorilla Logic Releases FoneMonkey 5

BROOMFIELD, CO.—Gorilla Logic has announced FoneMonkey 5, the latest version of the open source functional testing tool for iPhone and iPad applications. FoneMonkey improves application quality by automating the detection of bugs prior to release, shortening time to market, and reducing development and testing costs by lowering the need for resource-intensive manual testing processes.

FoneMonkey is the only test tool for iOS that records all actions with the iPhone or iPad while in use and plays them back as a test script at any time. It enables the interactive creation, editing, and playback of automation scripts that exercise an application's user interface. Using FoneMonkey, developers and quality assurance team members can create suites of tests that automate and perform user operation sequences and then verify results. FoneMonkey supports development as well as QA testing, and tests can easily be incorporated into continuous integration environments.

New features include:

- Robust script recording and playback—more comprehensive recording and playback for user interactions, including taps, keyboard input, dragging, and scrolling for all Cocoa Touch components
- Code generation options—FoneMonkey can now generate ready-to-run test scripts in Objective-C that can be run under OCUnit, or JavaScript that can be run with Apple's UI automation tools
- Optimized for both iPhone and iPad platforms and user interfaces
- FoneMonkey now works on both simulators and real devices
- Test playbacks can be completely automated

www.gorillalogic.com/fonemonkey

Coverity Integrity Control

SAN FRANCISCO, CA—Coverity Inc. has announced the release of Coverity Integrity Control, a new solution for code governance that enables software development organizations to set policies for code quality and security and then manage, monitor, and report on these policies as code is tested. With Coverity Integrity Control, companies can automatically manage and enforce standard code testing policies across in-house development teams, outsourced development teams, and software provided by third-party suppliers, gaining deep visibility into development risk across the software supply chain.

Coverity Integrity Control is an integrated solution that leverages code testing results from Coverity Static Analysis and offers advanced code governance features including: policy management, executive heat map alerts and code control panel, policy breach notification, third-party supplier SLA enforcement, and code testing and Coverity Integrity Manager Integration.

coverity.com/products/integrity-control.html

Cloud PaaS for PHP Developers

MOUNTAIN VIEW, CA and PORTLAND, OR—Under The Radar, PHP Fog has announced the general availability of its Cloud PaaS for PHP developers. PHP, the foundation of web-scale sites like Digg, Facebook, Photobucket, Wikipedia, and The White House, is a widely used programming language that represents a large portion of overall web development. PHP Fog's cloud hosting platform provides a reliable, scalable, fast PHP service leveraging all the latest cloud technology.

PHP Fog gives developers dedicated resources for their apps in the cloud and works seamlessly with existing PHP apps like WordPress, Drupal, Joomla, and frameworks like Zend, Cake, and Kohana. PHP Fog also features an app store for launching popular PHP apps like WordPress, Drupal, Magento, and SugarCRM.

www.undertheradarblog.com

Telerik Test Studio

WALTHAM, MA—Telerik has unveiled a host of new features and upgrades to Telerik Test Studio, the company's next-generation automated testing tools suite. This latest release focuses on improving overall productivity and ease of use, as well as supporting desktop test automation for Windows Presentation Foundation (WPF) and integrating with TeamPulse, Telerik's agile project management solution. With the debut of WPF testing capabilities, Telerik Test Studio, formerly WebUI Test Studio, now offers broader functionality extending from the web further into desktop testing. It also significantly expands the suite's reporting capabilities, greatly enhancing the overall user experience and making it one of the market's most easy-to-use testing solutions.

www.telerik.com/automated-testing-tools.aspx

FAQ

expert answers to frequently asked questions

by Robert Sabourin
rsabourin@amibug.com

How do you write good user stories?

Poker players talk about "tells" as the things other players do that give away the cards they are holding. Often, the question "How do you write good user stories?" is more of a tell than a problem.

A good story is about someone doing something interesting. A good user story is about someone doing something interesting with software or software doing something interesting for someone. In order to succeed with user stories, you need a few key ingredients:

Storytellers with Product Knowledge

User stories are the result of a group of people who realized that no matter how much we specify, people will ask questions. In a courageous move, this group asks people to shift from a mostly written tradition (e.g., formal specifications) to a blend of written and spoken tradition. If you are not connected to the person who can tell the story, your "tell" will be that you cannot write good stories, but this is not the source of your problem. Once you have the person (or persons) who can tell the story, writing the stories is simplified.

Users, Use, and Context

Once you have content, you need to provide context. While writing and telling stories, the audience needs to hear about user, use, and context. The context generators I use are:

Collaborative chartering: Builds the big picture that tells why you are doing what you are doing

Pragmatic personas: Descriptions of users and their interests (who's buying, who's buying in)

Story maps: User stories visualized in a way that tells the user experience or the big picture

As is normally the case, these are simply tools; you add the real value. Avoid getting lost in the recipe trap of following a prescribed process if it does not serve a purpose. For instance, many people write stories using the following starter template: "As a ___ I need to ____ so that___." Over time, this often kills storytelling and promotes sentence reading. When this happens, the narrative value of user stories dies and a slice of context dies along with it.

Testable Outcomes

As Steve Martin said in *Planes, Trains, and Automobiles*, "If you are going to tell a story, have a point." A last, but essential, step is: Express user value in testable terms. The best stories I see are those with the best story tests. The best tests are those that advocate on behalf of the user and drive toward real value and meaningful outcomes.

Keep your story titles short—around five words or fewer—and express the value in story tests. If you find you are having a hard time creating strong story tests, it may require some digging to ensure you have a good story to tell.

The best way to craft good stories ready for storytelling is to keep the story-writing ceremony low and the story titles short, and to focus on expressing real value in tests that speak to the value of a story and the value across stories.

Deliver Applications that Meet Business Needs

Concrete requirements definition is key to aligning software with business and user expectations. RDM is essential throughout the application lifecycle.

by Filip Szymanski | filip@hp.com

The world has become instant. In order for businesses to remain competitive, they must create new and innovative applications to meet changing needs.

One mistake people often make when building applications to meet business needs is not properly defining business requirements from the get-go. Poorly defined requirements compromise the quality of applications and lead to projects that deliver marginal value or fail outright.

It is imperative that requirements are well defined to avoid misunderstandings about their meaning or intent. Also, it is difficult to determine the dependencies of poorly defined requirements. A search function for a mobile application may depend on code for a related function in a browser-based application. If the requirements for the mobile application do not document this dependency, a change to the browser-based function may break the mobile application. If the requirements do not take into account these dependencies, unexpected surprises may wreak havoc on project schedules.

> "Even though the software is difficult to use, produces incorrect results, has some malicious side effect, or in some other way is not fit for use, it seems that "works as designed" is an acceptable response."

Even when requirements are well defined, a lack of proper requirements management may lead to major problems. For example, when requirements are captured using word processing or spreadsheet software, application teams are often unable to determine how a change to one requirement will impact testing, introduce new risks, or affect other parts of the application. With all this uncertainty, applications with poorly managed requirements take longer to deliver.

Requirements Definition Management

Concrete requirements definition and management support application development teams' delivering software that meets the needs of the business—on time and within budget. Throughout the application lifecycle, requirements definition management (RDM) should be a continuous process that includes eliciting, documenting, analyzing, prioritizing, and agreeing on requirements. Equally important, it entails controlling, managing, and analyzing the risk of changes to requirements.

Requirements definition includes:

Elicitation, in which business analysts gather requirements through interviews, questionnaires, user observation, workshops, brainstorming, use cases, role playing, and prototyping.

Elaboration, which adds more depth to requirements, breaking them down into tasks, technical details, or design specs. Methods used include developing use cases, flow diagrams, class models, and GUI mock-ups with business rules.

Validation, which verifies that requirements are complete and testable by checking for ambiguity, conflicts, or errors. Techniques used include discussions, simulations, and face-to-face meetings.

Acceptance, which occurs when the requirements have been verified and accepted by all stakeholders. Business analysts allocate requirements to releases so that technical development and test planning can begin.

Requirements management includes:

Requirements prioritization, in which the business and application team collaborate to prioritize requirements based on client need and business risk. This phase adds objectivity to the application lifecycle.

Traceability, which defines the association between requirements and other entities in the lifecycle, such as other requirements, tests, defects, and business processes. Linkages between requirements and other assets allow the business to track the status of an application to understand its true quality.

Change, which is the alteration, addition, or deletion of a requirement. The role of requirements management is to control and manage the impact of change so that all stakeholders in the lifecycle have visibility into the alterations made.

Roles and Responsibilities

Requirements are more accurately defined and business needs are better met when there is a joint effort between business analysts, software quality managers, and developers.

In the RDM effort, business analysts must define concise and unambiguous requirements. For each requirement, business analysts must write a succinct description in business language, establish the business value, quantify the risk, and document the dependencies.

Software quality managers must work with business analysts to ensure that the requirements can be implemented and tested. In their plans for testing the requirements, software quality managers must determine how best to assign resources to make the right trade-offs between business value and risk.

Developers should be involved in the requirements definition, review, and approval. They must verify that there is enough detail in the requirements to start development as well as set up a realistic timeline for development based on the complexity of the deliverable.

Stakeholders who implement strong RDM practices can increase application quality, align with the business, and deliver applications on time and within budget.

Waterfall Versus Agile

Some aspects of RDM can vary depending on whether the applications team is working in a waterfall or an agile environment. Most of the variation is in the timing for when each of the elements of elicitation, elaboration, visualization, and acceptance takes place.

In a waterfall environment, more time is spent up front on detailed requirements elicitation, elaboration, and visualization using simulations and business process modeling. In an agile environment, user stories (requirements) are defined in smaller increments with "just enough" detail to support each iteration. There are shorter bursts of elicitation and elaboration activity, while visualization is reserved primarily for more complex, higher-risk requirements.

In an agile environment, it is particularly important for all stakeholders to collaborate in the creation of user stories. Each agile iteration creates increased pressure to define user stories, estimate effort, develop, and test in a short period of time. It is this cooperation, continuous feedback, and quick turnaround that makes agile attractive. It enables IT organizations to rapidly deploy applications in response to changing business requirements.

Everyone Wins

If more focus on RDM is encouraged and implemented, projects are more likely to meet intended business metrics, including budgets, schedules, and client satisfaction. With RDM, all application stakeholders have a better understanding of what they need to accomplish, which leads to software that successfully aligns with business as well as user expectations. **{end}**

index to advertisers

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 © 2011 by Software Quality Engineering (340 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, 340 Corporate Way, Suite 300, Orange Park, FL 32073, info@bettersoftware.com.