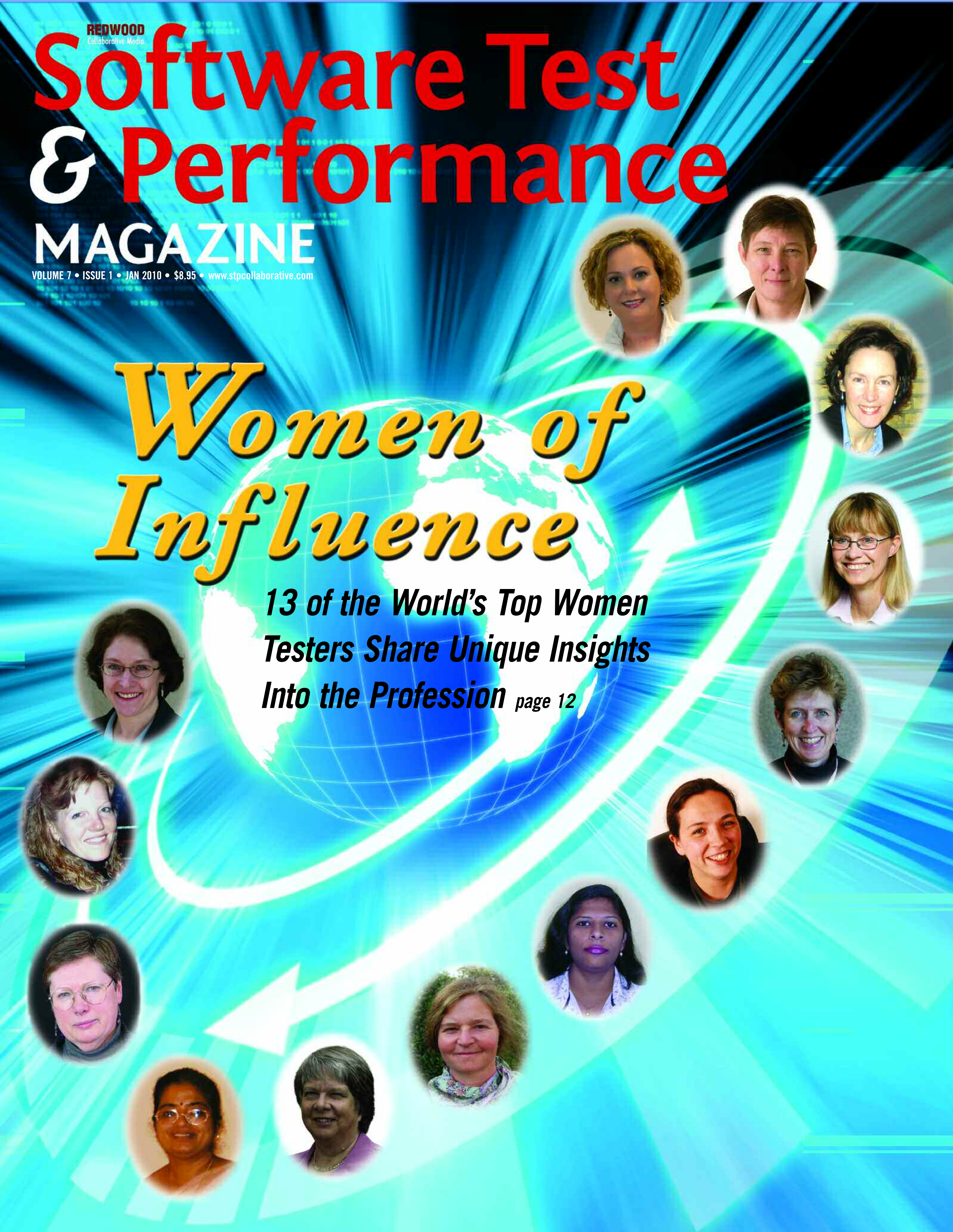
STARTUP TIPS FROM A TECH CEO | 6 WAYS TO IMPROVE AGILE PROJECTS

REDWOOD Collaborative Media

Software Test & Performance MAGAZINE

VOLUME 7 • ISSUE 1 • JAN 2010 • $8.95 • www.stpcollaborative.com



Women of Influence

13 of the World's Top Women Testers Share Unique Insights Into the Profession *page 12*

Join the leading community for the test & QA profession!

STP Collaborative serves a global community that provides more than 50,000 software professionals with information, education, training, & professional networking opportunities.

Become a member of STP Collaborative today!

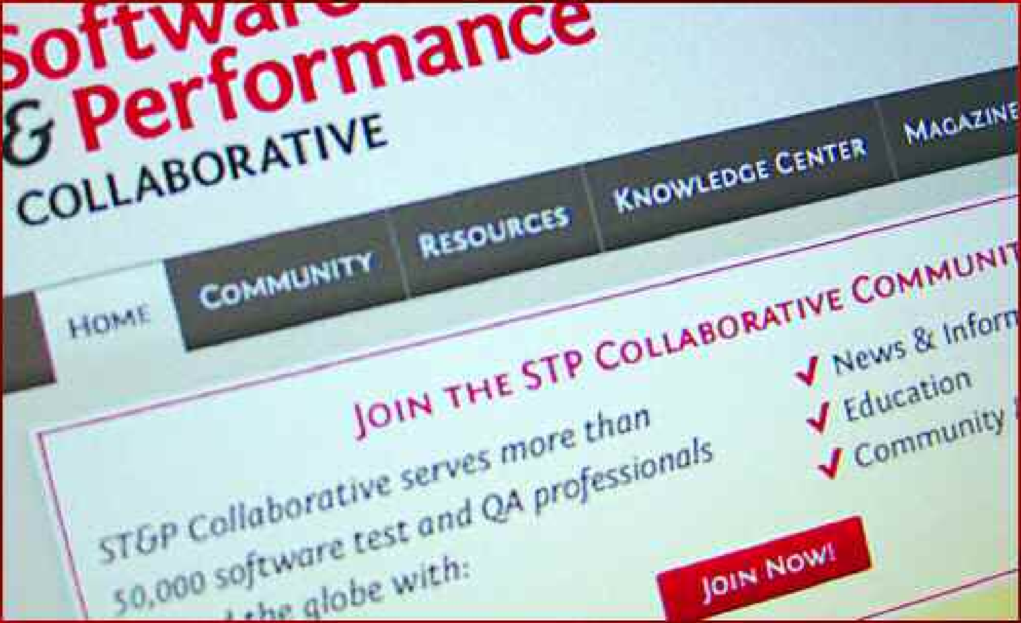
MEMBERSHIP BENEFITS INCLUDE:

- *Software Test & Performance* magazine
- Test & QA Report
- STP Insider eNewsletter
- Magazine & eNewsletter archives
- Resource directory
- Professional development opportunities: online & public training
- Member profiles
- Reader commentary
- Whitepapers
- Case studies
- Special industry reports
- Conference proceedings
- RSS feeds for news and blogs
- Member surveys
- Searchable content

...and more

www.STPCollaborative.com

Software Test & Performance COLLABORATIVE



JOIN THE STP COLLABORATIVE COMMUNIT

ST&P Collaborative serves more than 50,000 software test and QA professionals the globe with:

✓ News & Inform
✓ Education
✓ Community

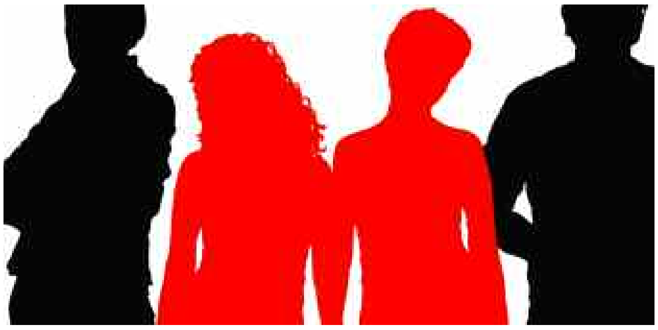
CONTENTS

VOLUME 7 • ISSUE 1 • JANUARY 2010

DEPARTMENTS

ON THE COVER

1. Elisabeth Hendrickson
2. Dawn Haynes
3. Fiona Charles
4. Renu Rajani
5. Dorothy Graham
6. Isabel Evans
7. Parimala Shankaraiah
8. Rosie Sherry
9. Lisa Crispin
10. Janet Gregory
11. Karen N. Johnson
12. Mieke Gevers
13. Sharon Robson



CONTRIBUTORS

FIONA CHARLES *(www.quality-intelligence.com) teaches organizations to match their software testing to their business risks and opportunities.*

KAREN N. JOHNSON *(http://www.karennjohnson.com) is an independent software test consultant based in Chicago. She is an international keynote speaker.*

LISA CRISPIN *(www.lisacrispin.com) is co-author of "Agile Testing: A Practical Guide for Testers and Agile Teams" (Addison-Wesley, 2009).*

JANET GREGORY *(www.janetgregory.ca) is co-author of "Agile Testing: A Practical Guide for Testers and Agile Teams" (Addison-Wesley, 2009).*

CATHERINE POWELL *(blog.abakas.com) writes about what she knows: teams, testing, and the everyday adventures of shipping software.*

LANETTE CREAMER *(lanette.creamer@gmail.com) is quality lead for Adobe Systems, where she coordinates cross-product testing for the Creative Suites.*

ELISABETH HENDRICKSON *(http://www.qualitytree.com/) is founder and president of Quality Tree Software, a consulting and training company.*

SHARON ROBSON *(SharonR@softed.com) is a trainer and consultant specializing in software testing for Software Education.*

NANCY KELLN *(nancy.kelln@shaw.ca) is an independent consultant with 12 years of diverse experience within the IT industry.*

ROSIE SHERRY *(rosie@schux.com) is a U.K.-based software tester turned social media professional and founder of the Software Testing Club.*

GET MORE ONLINE AT

Software Test & Performance COLLABORATIVE

ARTICLE

Check out the extended version of Publisher Andrew Muns' interview with dynaTrace CEO John Van Siclen at **stpcollaborative.com/vansiclen**

FORUM

In keeping with the theme of this issue, we'd like to hear about women who have influenced your career or the profession overall. Join us online at **stpcollaborative.com/womentesters**

STP BLOGS

How does testing Web 2.0 apps differ from testing earlier Web apps? Get in the conversation on Matt Heusser's blog, Testing at the Edge of Chaos, at **blogs.stpcollaborative.com/matt/**

WHITEPAPER

Download "Requirements-Based Testing: Encourage Collaboration Through Traceability" (sponsored by MKS) at **stpcollaborative.wufoo.com/forms/requirementsbased-testing/**

NEWSLETTER

STP's Test & QA Report offers commentary and insights on the latest topics in testing. See our most recent issue at **stpcollaborative.com/knowledge/519-motivating-test-teams**

RESOURCES

Research, source, demo and share your thoughts on the latest testing tools, technologies and services at **stpcollaborative.com/resources**

Software Test & Performance Magazine (ISSN- #1548-3460) is published monthly except combined issues in July/August and November/December by Redwood Collaborative Media, 105 Maxess Avenue, Suite 207, Melville, NY, 11747. Periodicals postage paid at Huntington, NY and additional mailing offices. The price of a one year subscription is US $69 for postal addresses within North America; $119 elsewhere. Software Test & Performance Magazine is a registered trademark of Redwood Collaborative Media. All contents copyrighted 2009 Redwood Collaborative Media. All rights reserved. POSTMASTER: Send changes of address to Software Test & Performance Magazine, 105 Maxess Road, Suite 207, Melville, NY 11747. To contact Software Test & Performance Magazine subscriber services, please send an e-mail to membership-services@stpcollaborative.com or call 631-393-6051 ext. 200.

And Now, The Women!

WELCOME TO THIS SPECIAL ISSUE OF STP Magazine, devoted largely to the theme: “Women of Influence in Software Testing.” It was conceived at the CAST 2009 conference in Colorado Springs. The July STP was just out, generating a happy conference buzz with its cover caricatures of 10 software testing stars and the accompanying feature article.

STP publisher Andy Muns spent that Monday afternoon in Jerry Weinberg’s tutorial. At some point the magazine got passed around. Tester Nancy Kelln said, “How come they’re all men?” As Andy told me later, he turned beet red, and had no immediate answer.

Into the expectant silence, Jerry said quietly, “Well, now you’ve done a feature on the men stars in testing, you’ll have to do one on the women. It’s only fair.”

So, here we are — thanks to Nancy, Jerry and Andy, who reflects, “Even though we didn't have any conscious bias toward men, those were the strongest relationships with the magazine run by a male publisher and editor at the time. We wanted to change that."

Since mid-October, when Andy engaged me as guest editor for this issue, I’ve pondered Nancy’s question: Why were there no women? And I have questions of my own.

- Why did it take a woman to ask that question?
- Why didn’t any of the featured men suggest women for the list?
- Did any of those men notice there were no women among them? Did it matter to them?
- Do men think women in testing aren’t good enough? Not innovative enough?

I don’t know the answers. Do you?

There are many smart, articulate women doing interesting and innovative things in software testing. Women lead major testing efforts and organizations. Women publish testing books, articles and papers; teach public testing classes; blog and tweet about testing; and facilitate and post on testing forums. Women initiate, organize and participate in social media and peer conferences. Women speak and conduct excellent workshops and tutorials at all the big testing conferences in North America, Asia, Europe and Australia/New Zealand.

Are women testers invisible? (Or not noisy enough?) I don’t think so. But in case you haven’t yet noticed our work, this STP issue brings you articles by a tiny selection of the many women who are doing and leading great testing.

You will find here a mix of established authors and new voices of women who blog but whose work has not previously appeared in print. I looked for a diverse range of authors representing different generations, testing philosophies and ideas, and geographies. We applied the same diversity principle in selecting women to interview for Karen Johnson’s “Women of Influence” cover story (see page 12), so you’ll find some overlap.

Powerful common themes emerge from the interviews and articles. These women testers speak from extensive experience in software projects in the field. They share a profound concern for people — for the customers they test for, and the colleagues they collaborate with, teach, manage and mentor. At the same time, they exhibit a balance between intellectual and analytical approaches to testing, creativity and pragmatism. Above all, these women are testers by choice. Their zest in the sheer joy of testing shines off these pages.

I had a lot of fun putting this special issue together. You may not agree with everything you read here — in fact, I hope you won’t! But you will be interested in, and engaged by, what these women testers have to say. Let us know what you think.

Don’t miss the other excellent content in this issue, too, including Andy Muns’ interview with dynaTrace CEO John Van Siclen.

This issue is dedicated to Jerry Weinberg, cherished friend, mentor and critic to innumerable software people — women and men alike — who was undergoing major surgery as I wrote this editorial.

Software Test & Performance MAGAZINE

VOLUME 7 • ISSUE 1 • JANUARY 2010

Editor
Amy Lipton
alipton@stpcollaborative.com

Contributing Editors
Joel Shore
Matt Heusser
Chris McMahon

Copy Editor
Michele Pepe
mpepe@stpcollaborative.com

Art Director
LuAnn T. Palazzo
lpalazzo@stpcollaborative.com

Publisher
Andrew Muns
amuns@redwoodcollaborative.com

Media Sales
Joe Altieri
jaltieri@redwoodcollaborative.com

Chief Operating Officer
Peggy Libbey
plibbey@redwoodcollaborative.com

Chief Marketing Officer
Jennifer McClure
jmcclure@redwoodcollaborative.com

Marketing Coordinator
Teresa Cantwell
tcantwell@redwoodcollaborative.com

Training & Professional Development
Abbie Caracostas
acaracostas@redwoodcollaborative.com

IT Director
Janette Rovansek
jrovansek@redwoodcollaborative.com

Reprints
Lisa Abelson
abelson@stpcollaborative.com
516-379-7097

Membership/Customer Service
membership-services
@stpcollaborative.com

Circulation & List Services
Lisa Fiske
lfiske@stpcollaborative.com

Cover Art by The Design Diva, NY

REDWOOD
Collaborative Media

Founder & CEO
Andrew Muns

Chairman & President
Peggy Libbey

105 Maxess Road, Suite 207
Melville, NY 11747
+1-631-393-6051
+1-631-393-6057 fax
www.stpcollaborative.com

Software Test & Performance MAGAZINE Announces its 2010 Editorial Calendar

Cover Issue	Cover Topic	Membership Focus	Ad Close/ Ad Materials Due
February	Testing Centers of Excellence	Testing Engineers/ Beginning Testers	12/30/09 1/4/10
March	Security & Vulnerability Testing	Financial Services	1/29/10 2/5/10
April	Virtualization & Testing	European Testing All-Stars	2/26/10 3/5/10
May	Web Load Testing	Consumer Web	3/26/10 4/2/10
June	Application Performance Management	Indian Subcontinent Testing All-Stars	4/30/10 5/3/10
July/August	Test Automation	Healthcare	6/4/10 6/11/10
September	Metrics & Measurement	Project Managers/CIOs/CTOs	7/28/10 8/3/10
October	Build management	Government & Defense	8/27/10 9/3/10
November/December	User Experience Testing	Testing & New Media	10/4/10 10/11/10

Software Test & Performance Collaborative reaches over 50,000 test and QA professionals through its magazine, newsletters and web offerings.

Promote your tools and technologies, products and services.
Contact Joe Altieri at jaltieri@stpcollaborative.com or (719) 476-0784.

Learn how you can contribute at www.STPCollaborative.com/contribute

10 Tips to Test Web 2.0 Apps

FIFTEEN YEARS AGO, OUR INTERactions with data on the Web weren't so different from our interactions with data on paper—publishing companies were beginning to post newspapers and magazines online, but video sites like YouTube and social media sites like Facebook and Twitter were barely a glimmer on the horizon. Now, though, our online interactions are way more sophisticated. Unlike the original, Web 1.0 model in which some "authority" — a media firm or retailer, for instance — provided information directly to users, Web 2.0 applications are created by and for users, who readily share content with each other.

So testing Web 2.0 systems, where the most critical functions involve interactions among users, provides some intriguing challenges. Such interactions are difficult to predict, especially when the number of users is large (think Facebook, with its whopping 65 million users). The value of those interactions is also tough to predict. For example, Twitter was designed for users to let their "followers" know what they're doing at any given moment. And while it does that, it also lets users create completely unexpected value.

What's more, Web 2.0 systems tend to include rich programming APIs that enable "mashups," user-created systems that integrate a particular Web 2.0 application into some larger system. For example, tagging a Twitter message a particular way can cause that message to show up on your Facebook page.

Based on our testing of numerous Web 2.0 applications, we offer the following suggestions for your Web 2.0 testing:

Matt Heusser and Chris McMahon are career software developers, testers and bloggers. Matt also works at Socialtext, where he performs testing and quality assurance for the company's Web-based collaboration software.

1. Transform your company into a staging area. In today's "enterprise 2.0" companies, communication among employees is essential. So before releasing new software features to your customers, try them in your own work environment. Not only does this help expose errors before the product ships, but it helps uncover issues with usability and value.

2. Log and monitor, monitor and log. The success of any Web 2.0 app depends on your understanding of what users do with the product — so you can give them more, better ways to do it — and where they might be frustrated by performance or usability problems — so you can make adjustments and keep users happy. To that end, have each user operation write itself to disk, including what was done and how long it took. Then, parse the logs to determine which operations are performed most and least often and how long each one takes.

3. Focus on what users focus on. In the old, Web 1.0 model, we needed to know how many hits each page received. With Web 2.0, we need more sophisticated models of user activity: Is the number of connections going up or down? Are the users creating small networks or big ones, or no networks at all? Are new features being adopted quickly or are they being ignored? Are users abandoning older features we should remove? By focusing testing on those features, you get customers to tell *you* what's important. This is a key distinction for Web 2.0 apps.

4. Play back the log as a performance test. This isn't a new tactic, but it's notoriously difficult to do well. The good news: New software architectures could make this sort of performance testing more feasible. In particular, if your application exposes a rich API that mirrors most or all of your business functions, it may be worthwhile to build clients that exercise the application according to the data in historical production logs.

5. Test in production. *Wait, let us explain!* If you group your code into versioned components, or modules, you can connect specific users to specific versions after login. For example, Amazon.com could have a list of power users, employees, and friends and family who run the latest, greatest/test/code, in production, after they log in.

Microsoft tests during production by rerouting such users — essentially, early adopters — to a special cluster after login; this cluster runs the latest production code. Thanks to server virtualization and cloud computing, this strategy is becoming increasingly viable, if still risky. (For details see *How We Test Software at Microsoft* by Alan Page, Ken Johnston and BJ Rollison; Microsoft Press, 2009.)

> "Before releasing new features to customers, try them in your own work environment."

6. Test with multiples users on one server and use special characters. If you're testing a Web server-based application by yourself, everything may work fine, but things often change when multiple users hit a piece of software at the same time. Work with another tester to simulate operations that might "step" on each other. You're probably already testing with !@#$%^ and other special characters as inputs, but the Windows utility *charmap* lets you insert UTF-8 characters. Use these characters as your username and anywhere values are entered. If the program supports internationalization, using foreign letters as a username is another quick way to test i18n.

Another technique, courtesy of Indianapolis tester David Christiansen (http://www.techdarkside.com/), is to paste *<script>alert("danger")</script>* and save everywhere input can be saved and redisplayed. If you see a "danger" popup on the next view, or you see the alert without the script tags, you've found a cross-site scripting vulnerability.

7. Automate tests to segment functionality. When a test drives the GUI, it's tempting to have that test do everything a human would do at setup and teardown, but the result, over and again, is an agonizingly slow test suite that's tough to maintain. Instead, test only the functions involved in using the particular feature you're testing, and make judicious use of export, import, SQL loads, copies of logs, command-line interfaces, batch jobs and so on. You'll end up with a much faster test suite that covers only basic functions. The only downside: You'll need other techniques to test the other functions.

8. Integrate your test suite into the application. At Socialtext, we built a wiki as an editable Web page. Our automated tests are expressed in a language called wikitext, which is generally human-readable. A wikitest to drive Google in a search for Matt's blog "Testing at the Edge of Chaos," for instance, might look like the test in Table 1.

TABLE 1: WIKITEST

open_ok	www.google.com	
wait_for_element_present_ok	search	30000
wait_for_element_present_ok	submit	30000
type_ok	search	Matt Heusser
click_ok	submit	
wait_for_element_present_ok	link=The Edge of Chaos	30000

We write these tests as wiki pages, and our test framework pulls the code from the wiki and executes it. This approach provides two major benefits: We can use our own application to create the tests, and nontechnical readers can access and understand the results from any Web browser. It's like living on staging (also known as eating our own dog food — or, as we like to think of it, drinking our own champagne.

9. Test on a production-data clone. It's tempting to run basic tests (creating users, accounts and various combinations) on a clean system, when a "dirty" system would provide more realistic results. We've been doing database testing for years on clones of production data, and we've discovered it makes sense to do the same on Web 2.0 apps.

10. Be ready to roll back. Build in hotswappable engineering capability so you can "roll back" new software to a previous good-state version in the event of a failure during production. If performance suddenly falls through the floor, a system administrator flips a switch and the software returns to a known good state. Sure, this is a reactive technique, but sometimes, better late than never! ☒

Index to Advertisers

Software Test & Performance MAGAZINE

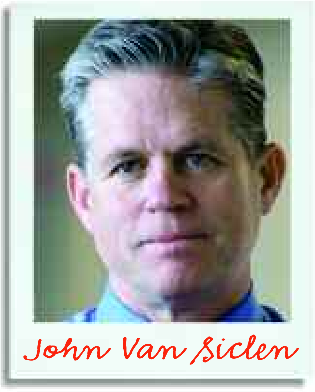
Advertiser	URL	Page
STP Collaborative	www.stpcollaborative.com	2- 3
STP Magazine	www.stpcollaborative.com/contribute	7
STP Collaborative	www.stpcollaborative.com	26
STP Collaborative	www.stpcollaborative.com/events	39
Hewlett-Packard	www.hp.com/go/alm	40

CEO VIEWS

By Andrew Muns

Don’t Stop at Innovation

AS CAPTAIN OF PRINCETON'S varsity hockey team in the late Seventies, John Van Siclen won the coveted Blackwell Trophy for sportsmanship. Thirty years later, he co-launched the New England Clean Energy Council. From sports to family to his newfound interest in greening the environment, the four-time CEO has always put passion into every endeavor. But nowhere does his exuberance shine more than in his role as chief executive of dynaTrace Software, the Lexington, Mass.-based, application performance management company.


John Van Siclen

Based on his experience leading tech startups, including Adesso, SupportSoft and Interwoven, Van Siclen shares insights into carving out a career path, developing management and team-building skills, and positioning products for fast growth in competitive markets.

ANDREW MUNS: How does a history major at Princeton end up in the IT space? Did you have any idea when you were a student that this would be your career path?

JOHN VAN SICLEN: Not at all. When I graduated, I was more interested in location than career. I was born and raised in California, so I ended up in the Bay Area. Through some Princeton connections, I ended up at a company called Qume Corporation, which was being acquired by ITT — at the time ITT was the greatest American business success story to date. That gave me a taste of technology as well as the high-growth startup environment right out of college, and it always stuck with me.

Those were the early days of the technology business, back in the late Seventies and early Eighties, when the sector branched from IBM into a number of companies and we went from the mini-computer business into a high-volume growth business segment.

So you caught the bug working at Qume and ended up working at a lot of other early-growth tech companies. What are some of the unique challenges associated with leading an organization in that phase of maturity?

One of the trickiest is to match the product with your go-to market. If the product is very difficult to implement, like SAP, you have to go after a very high-end integration channel, whereas, if something is really meant for a high-volume market, it better be extremely clean and easy to use, and it better work as advertised.

RESUMÉ

JOB:	*CEO, dynaTrace Software*
EDU:	*BA, Princeton University*
EXTRA CREDIT:	*Captain, Princeton Varsity Hockey Team; Founding Fellow, New England Clean Energy Council*

A lot of companies get that match wrong. They try to put the wrong kind of product through the wrong kind of channel and go-to market, and it tanks. My background has been in different channels to market and different business strategies to maximize the effectiveness and growth of a business by matching those two.

What are some examples where a company had what should have been a great product and a great idea that died because of that mismatch?

Take the Siebel/Salesforce.com battle. The way people wanted to buy sales automation tools was as a lighter-touch SaaS delivery model, but it wasn’t available at the time Siebel was dominant. The minute Salesforce.com made it available, it swept through and overtook the heavy-lifting product that took a year to ramp up and get a value out of, which was the old-style Siebel approach. Adoption of the new lighter-touch, faster time to value of a Salesforce.com was so rapid because the product matched the go-to market.

What life experiences outside the business realm helped build your leadership skills?

Probably two things. I happened to play a highly competitive sport that was also highly team-oriented, and also very unnatural for where I was playing it — I grew up in Los Angeles and played ice hockey. There are a few things that have always stuck with me from this. I don’t mind doing something that’s a little bit out of the ordinary. I’m highly competitive about it, and team-building is to me one of the key success attributes of any company.

The second thing is that by the time I was 30 I had spent as many years on the east coast as the west coast, so I have a blended personality—west coasters think I’m a highly intense easterner and east coasters think I’m a collaborative westerner. The combination of the two gives me a different perspective from many.

What do you think are the key attributes for a great leader?

I’ve had the benefit of working for some just tremendous ones and also working and seeing plenty of things to avoid. It comes down to a handful of things. One is to maintain the highest integrity. It sort of goes without saying, but there are many leaders who don’t have it, and those are the ones we don’t respect as much.

Another is having a clear vision and being able to communicate it. I know some great vision people, but they can’t communicate it. Other people are great communicators but don’t have vision. So that combination is very important, especially in early-stage companies where tight focus is fundamental to your success, as is how you measure and manage it to the goals you set.

Last, you have to have leadership

qualities. Those can sometimes come out over time. You can't shy away from tough decisions or from being out in front of the company and leading the way, in the field or inside your walls. You just have to have some of those characteristics.

What are the most important things to focus on as CEO of a software company in managing that development process?

One is focus—and again, this is especially important in the early stage so you don't try to do too much too soon. Like anything, it's not the fancy plays that create a great team, it's the execution of fundamentals. You really need to know what your fundamentals are and execute on those well before you start to fan out. I have seen many companies try to get too wide too fast, and that's a recipe for disaster. Usually young companies that are most effective bite off something they can chew, do it really well and then add to it over time.

Second is to get the product out early to customers so you get feedback and react to it quickly. The first version of a product is not about trying to win the market, it's really for you to learn.

As you grow, expectations go up. And with that comes more regimen in testing the product, validating with customers, communication and the rest. So there's a maturity curve that all products go through and a process for that.

Third is to make sure you have a product team that can execute. And it's not just about innovation. New companies do need to innovate, but if they can't productize, the innovation is lost. I've worked with companies that had great ideas and great innovation, but focused too much on "R" and not enough on "D."

It really comes down to getting the right mix on your team between innovators and productizers.

How, as CEO, do you create a team-building atmosphere that fosters effective collaboration and communication?

It's not easy and it's not always well done. The two key folks are the CTO, who is usually a founder in our kind of business and the person who has the brainstorm or the idea; it's another person, usually the vice president of engineering, who is the one to productize.

The chemistry and communication between those two team members is fundamental to success. You find a lot of broken technology companies as a result of a breakdown there.

It's an executive manager's responsibility to manage that dynamic early. Once the team is in place, if they're talented and you trust them, the dynamic will be positive. Hiring the right team makes your job is a lot easier.

I'm not the best at determining technical talent. I can determine business talent, and I'm pretty good everywhere outside of R&D. R&D is a unique skill set where you really have to be able to drill down into not only what people know but how well they apply it.

Over the years I've been blessed with some great CTOs and some great VPs of engineering. I've also had my share of CTO challenges, so I've seen it from both sides and know how precious that chemistry is.

> “It's not the fancy plays that create a great team, it's the execution of fundamentals.”

Software testers often debate how to advocate to senior management for more resources. How do you measure the value of the testing phase of the development lifecycle and allocate resources among development and testing?

The way I look at it is, an unhappy customer or a problem that manifests itself in the field is at least a hundred times more expensive to fix than if I found it and solved it back in development in the first place. And I do not think I'm unique in this view — I think most senior executives and CEOs know this. In fact, it's not even a question. The problem comes sometimes in the battle for resources between the production guys who are firefighting and the preproduction folks who aren't firefighting, especially if they're not very efficient or they're not providing full coverage. There needs to be a conversation internally about prior proper planning to prevent poor performance. And the preproduction side is really about prior proper planning.

To me, in building a software company, it's fundamental. To an operational company that is a distributor of technology products, or a retailer of XYZ, or a logistics company, it's not necessarily natural to think like software executives who know their business is all about their software. They're the ones who often look at the preproduction side and say, "There's a bunch of developers, I'll pay for them, but I don't want to pay for a lot of systems to make them efficient," or, "I'm going to put all my money in my production operations area because that's where the heavy lifting and big applications need to live." I think a lot of the dilemma that's out there for your readers is how to raise the awareness of management that prior proper planning actually does pay off.

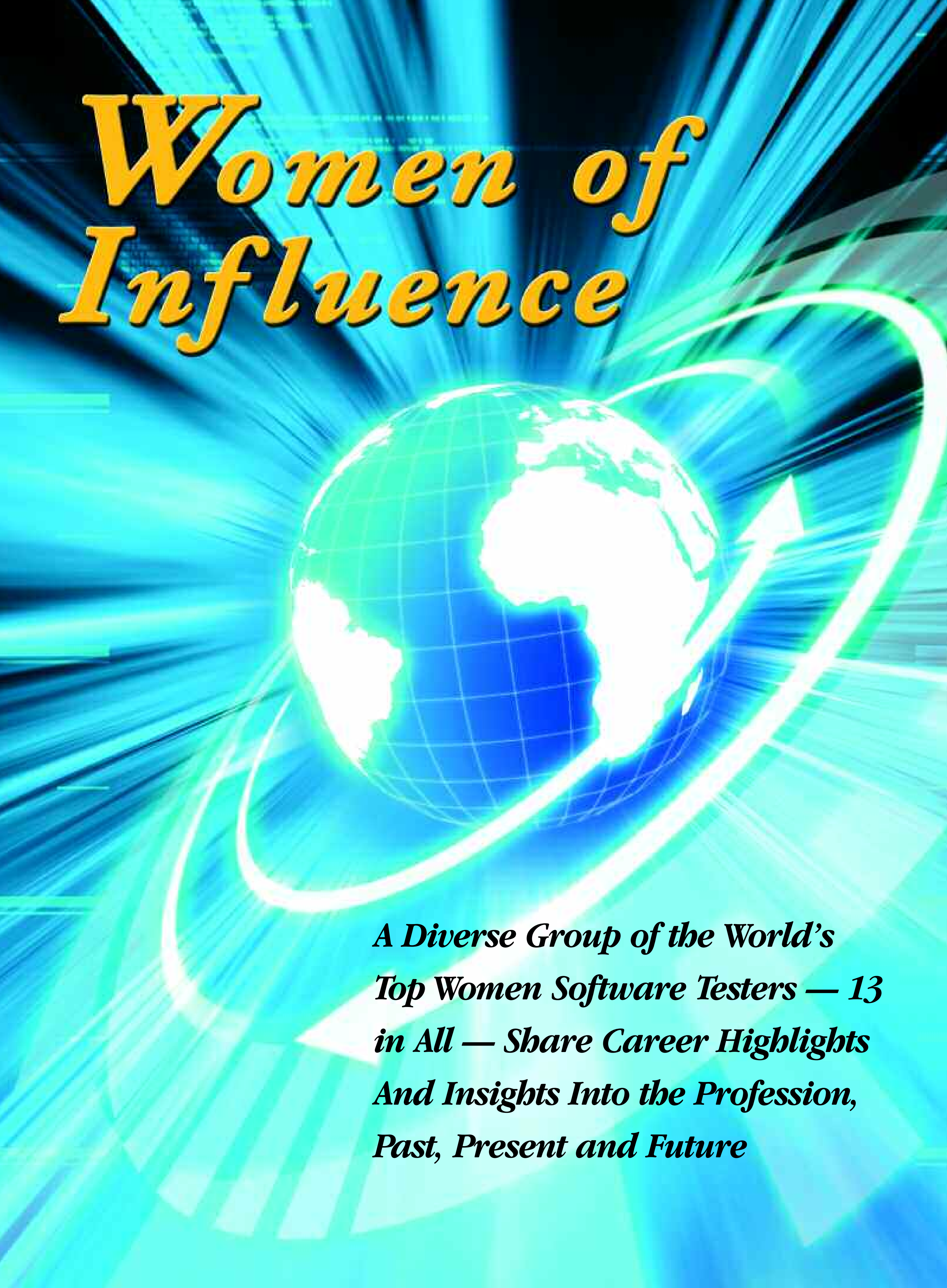
Again, when I talk to software companies — SaaS companies and the rest — they get it. It's when you talk to global 10,000 companies, they can struggle a little bit with it. What I've found to be most effective is just asking them the simple questions: What does an hour of downtime cost, how important are these applications to your business? How important is your reputation to you? How many users are affected by a glitch and how many of your customers are affected?

That usually spins the attention around to questions like, what do I need to make sure I test better, increase coverage and avoid shoving things to production to hit a date. In fact, one of the sayings we have at dynaTrace is, if you want to go fast, go slow up front — basically make sure you're really ready to run. ☒

FOR MORE with John Van Siclen, go to ***www.stpcollaborative.com/vansiclen****.*

In keeping with the theme of this issue, STP is interested in interviewing women CEOs of testing companies. If you know an exceptional woman CEO you'd like us to talk to, please contact Andrew Muns at amuns @redwoodcollaborative.com.

Women of Influence



A Diverse Group of the World's Top Women Software Testers — 13 in All — Share Career Highlights And Insights Into the Profession, Past, Present and Future

By Karen N. Johnson

Measuring influence is like capturing fog in a jar — virtually impossible. Sometimes influence is simple — a singular life- or career-altering idea or experience. Other times, influence is elusive — we can't pinpoint it on a map or calendar, we can't predict or repeat it. And what influences some people doesn't necessarily influence others. We're all exposed to a steady stream of perspectives from those in our professional lives — from co-workers and managers in our own organizations to peers and consultants we meet at conferences, workshops and training sessions. We absorb even more from industry publications, Web sites, blogs and local meetup groups.

But what sticks? What ideas and behaviors resonate with us, and which ones impact the way we do our jobs, pursue our careers, run our lives?

In the following pages, we speak with 13 women who've influenced the profession of software testing in various ways, both tangible and intangible.

The software testing community is large, and indeed, there are women testers the world over — so our list is diverse. It's not based on any formal process, nominations or rankings, and it's not limited to those whose names and faces are well known in the industry. It simply represents women guest editor Fiona Charles and I feel have contributed to the profession in some significant way.

We recognize that the list is short — *too* short; we had to limit it as a function of time and space. And, at the staff editors' insistence, it includes interviews with Fiona Charles and me. The following conversations appear in alphabetical order.

FIONA CHARLES

Company: Quality Intelligence
Position: Owner and Principal Consultant
Location: Toronto, ON
Web Site: www.quality-intelligence.com

Fiona Charles edited "The Gift of Time" *(Dorset House, 2008). She is the guest editor for this issue of STP. (See her editorial, page 6.)*

Karen N. Johnson *(http://www.karennjohnson.com) is an independent software test consultant based in Chicago. She is an international keynote speaker and has published numerous articles on software testing as well as recorded Webcasts.*

Photograph by Black Jack 3D

Fiona, tell us about your career and accomplishments in software testing.

I find it difficult to measure accomplishment — and especially influence — over 30-plus years. I can say what's on the resume. I've done project work in a tremendous range of software development environments: human, technical and business. With each new project, I've built my toolkit of models, heuristics, questions, patterns, techniques and strategies. I've been a technical writer and a director of QA, as well as a tester and test manager. I've managed projects and founded and run a testing practice. I've worked for two software product companies and two consulting firms, and now have my own company. I've observed, learned, adapted, made mistakes and solved problems.

FIONA CHARLES

[Our business is people. That fundamental ethic drives everything I do as a tester, manager and consultant.]

— Fiona Charles

I know I've played an important part in the success of many difficult projects. My work apparently influences people, but that's often hard to see at the time. Years later, I'll hear "Fiona taught me to write," "to see the big picture" or even "to speak truth to power." Those are humbling words — but ultimately the human interactions are more profoundly satisfying than what is on the resume.

What is your personal view of software testing?

Our business is people. Testers work with people and for people. Teams build software for people. That fundamental ethic drives everything I do as a tester, as a manager and as a consultant. It's all about people.

I fell in love with software and software development in my first IT job. I am endlessly enchanted by the process of building from logic an application that actually has the potential to improve people's lives.

Testing is solving problems. How do we find the software bugs and gaps that could hurt people the most? Striving to do that within project constraints is intellectually demanding and creative, both humanly and technically: from modeling the strategy to staffing and managing the team; from constructing test data to following your instinct that something in this feature smells wrong and there are bugs to be found — *here*!

What fascinates me about software testing is that the context in which we solve problems is continually changing. Each application, environment, business, project and team is different, and each demands a fresh approach. I love diving into a new project, immersing myself in everything I can absorb, while maintaining a critical tester's eye. What am I seeing, hearing, sensing? What have I missed? That's true whether I'm managing a testing project, or consulting with an organization to raise its testing capability or rescue testing that's missing the mark.

We can't impose predefined ideas, or

"processes," on a client. We have to observe each organization and its unique opportunities and risks. We have to listen to its people. I've always operated according to context and I've always been a systems thinker and a pragmatic problem solver. When I first heard there was a "school of context-driven testing," I was incredulous. What's the big deal? Why do we need a "school"? I couldn't join it — any more than I could bring myself to join a political party. I'm most in sympathy with the context-driven folks, but I can't be a card-carrying anything. That doesn't work for me or my approach. And I don't believe it would work for my clients.

LISA CRISPIN

Company: ePlan Services
Position: Director of Engineering
Location: Denver, CO
Web Site: www.lisacrispin.com

Lisa Crispin has co-authored two books:

- "Testing Extreme Programming" *(with Tip House; Addison-Wesley, 2002)*
- "Agile Testing: A Practical Guide for Testers and Agile Teams" *(with Janet Gregory; Addison-Wesley, 2009)*

She also contributed a chapter to "Beautiful Testing: Leading Professionals Reveal How They Improve Software" *(O'Reilly Media, 2009). (See the article on agile projects she co-wrote in this issue, page 20.)*

Lisa, tell us about your career and accomplishments in software testing.

Helping the agile and testing communities figure out how testers can contribute to agile teams, and how agile teams can best accomplish testing, have been my greatest joy. I promoted good ideas from other people, such as example-driven development and the agile testing matrix from Brian Marick, story test-driven development (which I think is originally from Joshua Kerievsky, at least his article is the first I read, and lately has been advanced by people like Elisabeth Hendrickson, Gojko Adzic and Antony Marcano), and collaborative test tools such as FitNesse. I have tried to pay forward all the help I got early on from the incredibly generous agile community.

Cowriting *"Testing Extreme Programming"* with Tip House and *"Agile Testing"* with Janet Gregory are the contributions of which I'm most proud. We wrote these books to share experiences and ideas we think will help other people, and based on the feedback we've gotten, we achieved our goal. I'm also proud to be a part of a good company whose top priority is quality, and feel that my team and I make major contributions to the success of the business as a whole.

What is your personal view of software testing?

LISA CRISPIN

ISABEL EVANS

[Passing on knowledge to the next generation [of testers] is important.]

— Isabel Evans

Use the "whole-team approach." I learned the whole-team approach from the early XP gurus, including Kent Beck, Ward Cunningham and Ron Jeffries. Everyone involved with delivering software takes responsibility for all testing activities needed to ensure the highest possible quality. Every team member has equal value. Everyone on a development team is a developer, including testers — and all developers do testing. Testing and coding are part of one process, not separate phases.

I've also been heavily influenced by Mary and Tom Poppendieck. Everyone on the development team (which includes the testers) learns the business. This allows us to help the customers make good decisions and contribute our own ideas to help the business improve. When I first started on an XP team, I read an article by Alistair Cockburn in which he said something like, "Software projects succeed when good people are allowed to do their best work." It's so true.

ISABEL EVANS

Company: Testing Solutions Group
Position: Principal Consultant
Location: London, England
Web Site: www.testing-solutions.com

Isabel, tell us about your career and accomplishments in software testing.

The quality improvement and preventive work (IT and across the business) is what I am most proud of, particularly in terms of what I did at K3 Group as the company quality manager from 1987 to 1991, where we managed to introduce flexible, adaptable, team-focused processes across the company, including design-your-test-first processes for development and strong customer involvement throughout the lifecycle. These are still the ways of working that I support.

I have also taken part in many software testing projects over the years as a tester and as a test manager, and I believe my work has been valued by project managers and developers. Now most of my work is training and consultancy, with some projects, but also activities that will help in retaining and passing on knowledge to the next generation [of testers], which is important. That is why I have tried to contribute to standards working parties and syllabus working parties, as well as speaking at conferences.

It has been a privilege to be recognized by the industry in a number of ways [and especially] being approached to write a book of my own. *"Achieving Software Quality Through Teamwork"* [Artech House, 2004] is a distillation of what I had seen working throughout my IT career.

My contribution has been very small when I compare it with those of others in the industry, but I have been able to introduce and use some positive ideas from outside the IT industry to benefit people and teams with whom I have worked, so I think it has been useful.

What is your personal view of software testing?

Software and software testing is in its infancy, so it's important that we cooperate across the IT industry to improve. Society as a whole needs better — but more invisible — IT systems. By that I mean, if IT is working, no one will notice it. Testing is only one way to achieve that, and not necessarily the best way, so we ideally should concentrate on improving software quality management — software development and maintenance processes and defect prevention. Testers need to cooperate with the rest of the IT industry to achieve that.

We also need to concentrate on the people who buy and use the IT systems — customer, business, society — as the stakeholders. What qualities do they require in the IT that supports their lives? Testing (the activity) needs to focus on checking the technology that supports the people who are affected by it.

Testers don't necessarily need to be specialist-independent testers. Other people (developers, analysts, project managers) can do great testing, and are naturally there early in the SDLC [software development lifecycle]. So helping those people do the testing can be very effective. It may be that the future role for specialist testers in many organizations is coaching and mentoring people in other disciplines rather than doing testing.

[Tools should be the vehicle to reach a well-defined goal, like cost reduction and better coverage.]

— Mieke Gevers

MIEKE GEVERS

DOROTHY GRAHAM

MIEKE GEVERS

Company: Aqis
Position: Director and Principal Consultant
Location: Belgium
Web Site: http://www.aqis.eu/

Mieke, tell us about your career and accomplishments in software testing.

Being in the IT testing industry for more than 20 years, I was able to taste from a wide range of aspects in testing. Working with automatic test tool manufacturers, like Rational SQA, Segue Software and Borland, gave me the opportunity to develop a special interest in the techniques and processes relating to performance management and automated testing, with its different facets, approaches, issues and solutions.

During the last two years at Segue Software, I had the chance to contribute to the product roadmap, always looking toward the "future of testing," which always fascinates me.

One aspect of testing — namely, performance testing — which can be so simple and yet so complex, is one of my favorites. The level of complexity we are dealing with day by day really intrigues me; out of all my notes collected over the years of hands-on, a performance testing methodology emerged. Another aspect gaining my attention, and whose growing importance is visible in the market, is agile and performance testing. Although the iterative development process approach was published by Victor Basili [*"Structured Programming,"* edited by V. Basili and T. Baker, IEEE Press] in 1975, it seems to us to be new and innovative. Nevertheless, it does bring up some questions for me: How can we as testers fit into this process? Which tools should we use? How can we do better? And many other questions I would like to see answered.

I'm trying to contribute where I can; by being a regular speaker at conferences, joining different committees, being a Eurostar country coordinator for Belgium and a Program Committee member of Eurostar 2007 and 2009.

Belgium did not have a formal testing community, so in 2006, together with three friends/testing colleagues, I cofounded the Belgian Testers Organization, and have since become a board member of KVIV and joined the BNTQB [Belgium and Netherlands Testing Qualifications Board], part of ISTQB [International Software Testing Qualifications Board], responsible for the Examination Workgroup. At all times I try to share this knowhow with others, as a coach, mentor, speaker, trainer or doing the hands-on and by staying a technician in heart and soul.

What is your personal view of software testing?

Tools should be the vehicle to reach a well-defined goal, like cost reduction, easier and faster testing, better coverage, etc., but they never should be the means itself. Don't forget, "A fool with a tool remains a fool." Also, think of the application's performance like response times, from the beginning of the development lifecycle — yes, even in Agile. Starting performance testing early will give an advantage to everyone.

DOROTHY GRAHAM

Dorothy has co-authored three books on software testing:

- "Foundations of Software Testing: ISTQB Certification" *(with Erik Van Veenendaal, Isabel Evans and Rex Black; International Thomson Business Press, 2008)*
- "Software Test Automation" *(with Mark Fewster; Addison-Wesley, 1999)*
- "Software Inspection" *(with Tom Gilb; Addison-Wesley, 1994)*

Dorothy, tell us about your career and accomplishments in software testing.

When I first decided to specialize in software testing, most people thought I was a bit mad! In the 1980s, a career in testing was unheard of; there were only a few books on software testing, one nonacademic testing conference (in the U.S.), no qualifications in testing and most managers didn't want to discuss testing.

My first job was as a programmer in a test group for Bell Labs in New Jersey — this is how I first got into testing. I wrote testing tools that became some of the earliest "shelfware," but I did get interested in testing. In my work as a developer at

Ferranti, I put testing “on the map” for the projects I was involved in (police command and control systems).

I've been privileged to have been involved in helping to raise the profile of testing from a widely perceived “necessary evil” (sometimes not even considered necessary) to a respectable profession (or at least a recognized career choice). I wrote training material in software testing for the National Computing Centre in the U.K., another training company (now gone) and clients (including Unisys).

In 1993 I was program chair for the first software testing conference outside the U.S., and was delighted to be program chair again this year.

During the 1990s, I also authored and co-authored four editions of the *“CAST Report,”* which described all known commercial tools that supported testing.

With my former colleagues at Grove Consultants, I trained thousands of testers. Grove continues to use training material I helped to develop. What I enjoyed most about training was seeing people’s minds change. I have been accused of “making testing interesting” – but of course it already is interesting, I just helped people to see that.

I am pleased to have worked on getting tester qualifications started in the U.K., and spreading worldwide. Achieving a qualification does not mean you are a great tester, but at least it can remove the bottom layer of ignorance about what testing is for many people.

What is your personal view of software testing?

Testing is great fun and very challenging, and encourages curiosity, deviousness and destructive tendencies. Thinking of what to test — that is, test design — is therefore best done by people. If someone thinks testing is boring, he or she is not thinking of test design, but of executing tests, often the same tests over and over again. Test automation is useful because we can get computers to do these things that are tedious and error-prone for human beings to do. It is more important to do good testing (design good tests) than to automate whatever tests you have now: Automated chaos is just faster chaos.

JANET GREGORY

Company: DragonFire
Position: Agile Coach and Process Consultant
Location: Calgary, AB
Web Site: www.janetgregory.ca

Janet Gregory co-authored “Agile Testing: A Practical Guide for Testers and Agile Teams” *(Addison-Wesley, 2009) with Lisa Crispin. (See their article on agile projects in this issue, page 20.)*

Janet, tell us about your career and accomplishments in software testing.

My greatest accomplishment in the world of software testing is sharing my experiences on teams transitioning to agile development methods. There is no secret one way, there is no one great achievement; it is the collection of years of experience. I take these experiences and form them into something that people can understand and use in a practical way.

By sharing my experiences, both good and bad, I try to help testers understand their role on an agile team. The book I co-authored with Lisa Crispin addresses many of my thoughts to a wide audience, but it is working directly with testers on an agile team that gives me the greatest pleasure.

I like to see when testers understand the skills they have developed over the years are useful not only in the traditional fashion, but also on agile teams. I try to instill confidence in other people so they recognize that, as part of an agile team, their role is to question, challenge assumptions and guide development. Being part of an agile team means growing and learning new skills, like automation, so they have time to use their intuition and testing skills in exploratory testing.

Many of the testers I talk to tell me they are still fighting a "throw it over the wall" attitude in their development teams. I try to give them a different way of looking at testing so they will keep on testing.

What is your personal view of software testing?

Testing only software that already exists is too late in the game. Moving testing forward to challenge assumptions and help guide development takes a lot of guesswork out of the game.

JANET GREGORY

DAWN HAYNES

[I’d like to tell all the managers and employers of testers to walk a mile in our shoes.]

— Dawn Haynes

DAWN HAYNES

Company: PerfTestPlus
Position: Senior Trainer and Consultant
Location: Palm Bay, FL

Dawn, tell us about your career and accomplishments in software testing.

The accomplishments that mean the most to me are the ones that involve being a "tester advocate.” We testers are not part of one established/consolidated community. There is no universal foundation of information we all learn before getting our first testing job (or even during our first testing job); frequently our managers and executives don’t understand what we do, how to manage us or even how to ask us for the information they really need, and there is really no place for everyone to go to get help. That means, for the most part, every test team is an island unto itself, and every tester is an individual who travels from island to island figuring it all out more or less on his or her own throughout his or her career.

I know I can’t change that situation. I’m not even convinced that testers could change much of it if we all banded together in a well-organized, grass-roots movement to improve things. What I believe I can do, and try to do every time I teach a class, speak at a conference, write an article or just run into testers “in the wild,” is help testers realize they are not alone, share with them the best of the tips, tactics and techniques I've acquired as I stumbled through my own career, and generally try to enable testers to be just a little more effective and a little happier when they get back to the office — one tester at a time.

What is your personal view of software testing?

I'd like to tell all the testers in the world that they're not alone. That most of the challenges they are facing are really the same challenges most everyone else is facing. That what we do really does make a difference most of the time. And to remember to not take business decisions by managers and executives to do the opposite of what you recommended personally; most of the time, they really are trying to make the best decisions they can for the company as a whole based on information we're unlikely to ever be aware of.

I'd also like to tell all the managers and employers of testers in the world who have never been testers themselves to walk a mile in our shoes. To experience, firsthand, the challenges, complexities, ambiguities and frustrations we run into almost every single day. To honestly learn what we can do, not just what we've been assigned to do, for them. And conversely, to be open and candid with us about the logic behind the decisions they make that clearly appear nonsensical to us.

To put it another way, I'd like to tell everyone directly or indirectly involved with software testing to do their part to get the whole team in the same boat working together to get that boat safely and efficiently to the same destination. I know it sounds like a little thing, but I can think of no single thing that is more valuable to a test team and the project it serves than having every single member of the team on board with the same mission, goals and priorities from kickoff to handoff.

[Testing that does not provide value to the business is waste.]

— Elisabeth Hendrickson

ELISABETH HENDRICKSON

Company: Quality Tree Software
Position: Owner
Location: Pleasanton, CA
Web Sites: http://www.qualitytree.com/ and http://testobsessed.com

Elisabeth, tell us about some of your accomplishments in software testing.

The test heuristics cheat sheet is a PDF download available for free from my Test Obsessed site, and it's been enormously popular. People from all over the world have told me they have printed it out and have it hanging in their workspace. When I created it I just wanted to make a little one-sheet reference for some of the test design heuristics that I teach in my software testing classes, but it's turned out to have far greater lasting value than I ever imagined.

ELISABETH HENDRICKSON

KAREN N. JOHNSON

Also, my green Test Obsessed wristbands. I got the idea of getting wristbands made when my youngest daughter came home from a summer camp wearing a wristband emblazoned with the name of the camp. It's how the camp identified the kids when they went on field trips. So when I created the wristbands, I thought they were a cute way to signal an obsession with testing — I didn't think anyone would take them too seriously. I was wrong. I've been absolutely delighted to discover that, for some, the wristbands are much more than an 8-inch circle of green plastic; they symbolize a deep commitment to testing.

My other contributions are less tangible. I've had a hand in organizing several small peer-driven workshops, including the Agile Alliance Functional Testing Tools workshops, and I'm delighted at how such events foster a spirit of community and collaboration. I also spend much of my time teaching and speaking.

What is your personal view of software testing?

Professional testers are in an excellent position to provide a tremendous amount of value to any software project by providing insight into vulnerabilities and risks. But testing that does not provide value to the business is waste. All that time spent hunting down a bug that no one cares about and no one will fix? Waste. Time and money spent on test automation that adds no value to the project? Waste. Holding a project hostage until it passes an arbitrary quality bar that has nothing to do with the actual business goals of the project? Not just waste, but actively damaging to the business. To ensure that we actually provide all the value we can, we must seek feedback from the business on an ongoing basis to make sure the testing effort is in line with the business goals.

(See Elisabeth Hendrickson's article *"The Politics of Testing: Making Conflict Count,"* on page 30.)

KAREN N. JOHNSON

Company: Software Test Management
Position: Owner
Location: Chicago, IL
Web Site: www.karennjohnson.com

Karen N. Johnson contributed a chapter to the book "Beautiful Testing: Leading Professionals Reveal How They Improve Software" *(O'Reilly Media, 2009). She was interviewed for this article by STP guest editor Fiona Charles.*

Karen, tell us about your career and accomplishments in software testing.

At this point in my career, I feel my writing about software testing is my primary contribution to the field. My blog is frequently focused on how I feel about my work, my thoughts and reactions to testing even more so than on testing itself or tactics and techniques. My articles are frequently filled with stories based on my work.

When I wrote the chapter in *"Beautiful Testing,"* it just poured out. My chapter is a story and the story is extremely personal. Software testing is not just a profession for me – it's personal and it is much of my life. Not surprising, my writing sounds that way.

My career at this point has a mix of activities, a variety I had longed for. I build classes and teach, write articles, speak at conferences and continue to have hands-on time testing on projects. I frequently tell people, I feel I have one of the best jobs. I love the work I have in front of me — especially the mix. Admittedly, independent consulting has its share of stresses and uncertainties, but for someone who likes change and variety, it can be quite interesting.

I've not been alone by any stretch. I've met and learned from so many people in our field over the years. I am a perpetual student, still learning, reading most nights. Cem Kaner and James Bach are two people who have strongly influenced my learning. Rob Sabourin and Mike Kelly are two people I spend as much time with as I can. I could mention other people, but there is a word count I have to keep in mind! I think my colleagues would agree the biggest thank you I could give would be to contribute back to our field, to give back to the community.

What is your personal view of software testing?

I envision the community of software testing as a large table and around that table there are seats filled with many different people. Each person brings a unique background, ideas and, of course, opinions. We're together in the sense that we all have a seat in our profession, we all bring something to the table. We are united by our profession, even though, like a family seated at a dinner table, we disagree — sometimes strongly — but then we're people, after all!

I see the community spreading across not just America or North America but around the world. A very large table filled with different philosophies, approaches, and experiences and of course, different cultures and backgrounds too. I like and appreciate diversity even when I don't agree with the other person or when I can't seem to apply other people's techniques to my own work. When I attend software testing conferences, I value the opportunity to meet other testers, even when our philosophies don't align, to hear first-hand what people's beliefs and experiences are. We are united at least on some level by the craft of software testing.

[How to provide best coverage with optimal test effort is both science and art.]

— Sharon Robson

RENU RAJANI

SHARON ROBSON

RENU RAJANI

Company: IBM India
Position: Delivery Executive
Location: Bangalore, India

Renu, tell me about your career and accomplishments in software testing.

I co-authored [with Pradeep Oak] the book *"Software Testing: Effective Methods, Tools and Techniques"* published [in 2004] by McGraw-Hill. This was one of the first books on software testing in the Indian IT industry; today it is used in a number of engineering colleges and referred by many software practitioners. This work got me recognition as a "Testing Thought Leader in Indian Industry" at the Test 2008 Conference, organized by PureTesting, India.

I have built one of the largest and strongest test competencies for IBM in India, providing test services for a large base of IBM customers across geographies. I've been successful in creating test factory services and providing centralized test services for business units of a large financial institution. The concept of "factory" or "utility," while well understood and practiced for manufacturing (automotive, etc.) and service utilities (gas, water, electricity), is [only] now real in software test services.

What is your personal view of software testing?

Testing is a focused discipline, to be taken as a dedicated career, with all the passion it deserves. Those who see it as a "necessary evil" — as an activity or a stop-gap job — would not do justice to it.

Coverage of risks is key to successful testing. Test coverage is a key metric for test activity. How to provide best coverage with optimal test effort is both science and art. Understanding of domain and application knowledge can't be underestimated. Clients today recognize that a vendor can provide value in testing only with creation and retention of domain knowledge and insist on domain knowledge as a key capability for "core" teams in a core/flex model.

Testing is a risk-management activity. Testing can be endless if not smartly done. How deeply one tests depends on risks to be addressed and impact of those risks if uncovered in real life.

Testing is not a stand-alone activity to be done at the end of the software development lifecycle. The testing lifecycle needs to be aligned to suit newly emerging software engineering lifecycles — iterative, rapid application development, etc. Testing steps cover the entire software lifecycle and each step in the software lifecycle should include validation.

SHARON ROBSON

Company: Software Education
Position: Knowledge Engineer
Location: Brisbane, Australia
Web Site: www.softed.com

Sharon, tell us about your career and accomplishments in software testing.

My biggest accomplishments range

from the individual level through to the corporate and then global level. At the individual level, when I am working with a team or training a group of new testers, I love seeing the spark of knowledge grow in their eyes. I count that as a key accomplishment, as it means I have opened the world of testing up to others. I know then that the knowledge and techniques of testing have found another new home, and that these people will commit to better quality in all they do.

At the corporate level. I built a team of testers from a dispersed group who went on to have a common language, common techniques and excellent practices, which in turn convinced the rest of a somewhat cynical development team of the value of testing in the SDLC. The team knowledge building and team integration approach resulted in my inclusion as test manager for a very large piece of national government work, from the very beginning of the project, able to be involved and contribute to the structure of the entire project, helping focus the project team on the value of quality and the inclusion of testing at each step to drive toward customer satisfaction at all stages of development.

Another thing I'm proud of is my activity at the certification level in testing. I am a founding board member of the Australia New Zealand Testing Board (ANZTB) and the chair of the Marketing Working Group for the ISTQB, which means I am actively involved in the knowledge growth about testing and raising the profile of testing practices at a global level.

What is your personal view of software testing?

Testing is what it's all about. Ultimately, we have to prove that the system being built meets the customer's needs. Testing does that.

Testing is all about figuring out "what we're trying to find" and "how we'll know when we've found it." Based on these two thoughts, we can structure our testing to suit the needs of any piece of work.

"Software testing shows the value of the system that has been built — for the team it is empowering, it is validating, it is the rationale for the work that has been done and what is yet to come; for the customer it is the icing on the cake."

(See Sharon Robson's article *"The Power of Pessimism"* on page 32.)

PARIMALA SHANKARAIAH

Company: Consona CRM India
Position: Senior QA Engineer
Location: Bangalore, India
Web Site: http://curioustester.blogspot.com/

Parimala, tell us about your software career and about "weekend testing" in India.

I am one of the initiators of the Weekend Testers community in Bangalore, which was started with an objective of practicing testing on weekends. I have facilitated many weekend testing sessions where testers from different parts of India have participated in testing open source systems to improve testing, communication, note taking and facilitation skills. As part of Weekend Testers, I have helped provide a challenging setup for testers to practice exploratory testing and help the testing community.

What is your personal view of software testing?

Software testing is an amazing craft based on exploration, discovery, investigation and learning. I feel proud to be practicing this craft.

(For a more in-depth conversation with Parimala Shankaraiah about women software testers in India, see page 37.)

PARIMALA SHANKARAIAH

ROSIE SHERRY

[If someone were to say software testing is boring, I'd say think again.]

— Rosie Sherry

ROSIE SHERRY

Company: Software Testing Club
Position: Founder, Community Manager
Location: Brighton, England
Web Site: http://www.softwaretestingclub.com

Rosie, tell us about your software testing career and, in particular, the Software Testing Club.

Several years ago I started a blog on software testing. I was keen to get to know people in the field. Build my confidence up. I then thought it would be a great idea to start an online community for software testers — the Software Testing Club [STC].

I didn't think it would take off, but it did. It's been going for two and a half years now and constantly growing, not only in number of members but in quality of discussions. On the back of it, we are trying out some new ideas, too.

When I say "we," I mean myself and other members who have been core in getting the STC to where it is. It's not about me. It's about the testing community. It's about adding some "va-va-voom" to our industry, which is often excessively overshadowed by politics and corporate bureaucracy.

I wouldn't feel so proud without the STC. The positive feedback is lovely. And I know and hope that as a result of it other testers have benefited in unique ways.

What is your personal view of software testing?

If someone were to say software testing is boring, I'd say think again. You only need to meet a few passionate testers to find out that testing has a bright future.

Perhaps testing is now at a tipping point. There are testers out there who can make a difference. We just need to convince the rest of the world.

& ***We want to hear about women software testers who have influenced your career or the profession as a whole. Join our online forum (it's not for women only) at www.stpcollaborative.com/womentesters.***

Agile Projects: 6 Ways to Avoid the 'Mini-Waterfall'

By Lisa Crispin and Janet Gregory

Many new agile software development teams experience a common problem — the "mini-waterfall." They do two-week iterations, but each iteration includes a requirements phase, design phase, coding phase and testing phase, and testing may lag an entire iteration or longer behind the coding. Even if the team practices Test Driven Development (TDD) and does some Acceptance Test Driven Development (ATDD), the stories still drag out to the end of the iteration. Time to test the finished code is nonexistent, new features are unreleasable and testing activities (especially any GUI test automation) are pushed to the next iteration. Testing falls further behind, features must be reworked, there's no safety net of automated regression tests, and the team delivers less and less value over time.

Several techniques, discussed below, can be used to alleviate this problem of turning agile iterations into a microsized phased and gated process. Although each of these approaches individually will help you avoid the mini-waterfall, the real power is in having the entire team adopt them all:

1. Focus on one story at a time
2. Keep stories small
3. Collaborate
4. Reduce feedback cycles
5. Automate
6. Use the "whole team" approach

FOCUS ON ONE STORY AT A TIME

No user story is done until it's tested, so how do we get all the testing done before the end of the iteration to be sure our code is releasable? We have to spread the testing out over the entire iteration, and the best way to do that is to focus on finishing one story at a time.

Clearly, there's a limit to how many people can work efficiently on the same story simultaneously. Still, by focusing on finishing the highest-priority story, including its testing activities, and then the next highest-priority story and so on — and if we size our stories appropriately — we can have our first story fully ready for testing a few days into the iteration, with the next story to follow shortly.

Here's an example of how this works for a two-week iteration. Let's say our team consists of four programmers and two testers, and we've planned four stories. We'll call them A, B, C and D. They're arranged in priority order (corresponding, conveniently, to alphabetical order) on the storyboard, one story per row. The team, working with the customers, has written high-level acceptance tests for each story during iteration planning.

Day 1: Two programmers pick up tasks for Story A, and two start on Story B. The team is focused on finishing Story A, so both testers pair to review the high-level tests for Stories A and B and expand the test cases to share with the programmers to get feedback and guide them in their coding tasks.

Day 2: The testers automate the tests and get as far as they can with test tasks until some code is ready. They've had some conversations with programmers and customers about the stories, and refined the tests. Story A is well under way, but there's a UI task that still needs to be done. One of the developers on Story B picks up the Story A UI task so story can be finished. The testers expand on Story B tests to share with the programmers before they get too far in their coding.

Day 3: The testers start on Story B tasks in the morning, but by noon the happy-path acceptance test for Story A is passing. The testers switch back to Story A, writing more detailed automated tests and doing their manual exploratory testing. They also review the high-level tests for Story C so the programmers can start on it.

Lisa Crispin *(www.lisacrispin.com) and* ***Janet Gregory*** *(www.janetgregory.ca) are the authors of "Agile Testing: A Practical Guide for Testers and Agile Teams" (Addison-Wesley, 2009). Crispin has worked as a tester on agile teams for 10 years. Gregory is an agile coach, helping business users and testers understand their roles in agile projects.*

Photo by Jose Manuel Gelpi

Day 4: The team feels confident it has covered Story A with both automated and exploratory manual tests. The testers switch focus to Story B, and two developers start working on Story C.

Rinse and repeat: This rhythm of focus, finish, focus, finish continues until all stories are complete. The team stops all new development on Day 9, to allow Day 10 wrapup of testing and endgame activities and avoid last-minute scrambling. The last day or two of the iteration is sometimes devoted to final regression testing, extra-functional testing, such as load or security, and release tasks (see Figure 1).

Not everyone can work on one story, but when tasks remain to finish a story, everyone pitches in to complete them. If the end of the iteration approaches and there are lots of testing cards for unfinished stories, the programmers pick those up to get the stories finished in priority order.

By concentrating on finishing the top-priority story in progress, you ensure that at least some stories will be completely "done" by the end of the iteration, instead of having four stories that might have coding but not testing completed, or four stories that are 80 percent done and not ready for delivery.

KEEP STORIES SMALL

Big stories usually translate into longer delays for some testing activities. If it takes a week to finish any stories in the iteration, the team is sure to run out of time to complete all testing tasks. In addition, teams are more likely to underestimate the work involved in bigger stories.

Try this: Size stories as small or medium, where medium is no longer than three calendar days of programmer effort to code and unit-test, with one or more team members working on the story. If the story is bigger than that, ask the customer to slice it up into smaller pieces.

This isn't easy, especially if you're new to user stories. It's hard for customers to see any business value in a small story. Nevertheless, small stories have many advantages. You're less likely to have a story "blow up" on you if it's relatively small. Working on stories that take a day or two — three at most — helps the team get into a nice rhythm. Work — writing tests, writing code, testing some more — flows through the team continually. The team is more likely to finish stories at a steady pace throughout the iteration, without lots of testing tasks left over at the end.

If your stories are too big or complex, use a "thin slice" or "steel thread" technique to split them up. Start with an end-to-end path through the story, a thin slice that represents a minimum of core functionality. This keeps the team from getting stuck on, say, the first page of a four-step UI wizard, and ensures that the essential functionality of the story — including testing — is finished. We can enhance testability and remove integration issues as quickly as possible.

Say, for example, we have the following story: "As a customer on the Donkeys and Dragons Web site, I want to click on an account name in a list of my payment account names and add, edit and delete the account information." We could slice this up as follows:

Slice 1: Display the page that lists the payment accounts. The name is a link. Clicking on the link takes the user to a second page, which displays existing accounts from the database. There's no add, edit or delete capability yet, but the page has navigation back to the first page or elsewhere in the application.

Slice 2: The second page includes the ability to edit, and has data validation for the different fields, but changes don't persist to the database.

Slice 3: Changes on the second page persist to the database when the save button is clicked. In addition, the second page has reset and cancel buttons.

Slice 4: The user can add a new account; with all the validation, the new account is persisted to the database and redisplayed on the list.

Slice 5: The user can delete an existing account if it isn't used in an active order.

For each thin slice, the team writes and automates test cases, writes the code and does exploratory testing, building on it for each subsequent slice. Testing activities are spread out, so testers don't have to wait until all the code is finished. Even if the last slice isn't complete, the story may be releasable. If the delete functionality weren't finished, for example, the business could opt to handle deletes offline until the delete functionality could be put into production.

COLLABORATE

Collaboration among customers, programmers and testers is essential to finishing stories in a timely manner. The collaboration starts when the team defines the acceptance tests. When there's input and discussion by the whole team, the tests are richer and the understanding of the story intent increases. This interaction translates directly into building the right thing.

When testers share their tests with programmers, several things happen. First, it provides an opportunity to get input to the test design and test cases from the programmers, which makes the tests more robust. Second, it eliminates misunderstandings about the behavior or misbehavior of the functionality. This can save rework time later. Testers will find fewer unit-level defects, because the expectations have been set. Third, the tests can guide the programmer's coding, especially if they're automated. And last but not least, knowledge is shared — programmers learn more about testing, and testers learn more about risky areas of the code.

During the iteration, when the team works together to find and fix defects immediately, there's less time spent sending bugs back and forth through the defect tracking system. Less time is wasted trying to determine if something is a bug rather than a new feature, and more time is spent developing good software to meet customer expectations.

REDUCE FEEDBACK CYCLES

Agile development is all about iterative feedback loops. The less time spent getting feedback to those who need it, the less waste in the cycle. Earlier we mentioned fixing defects as soon as they're identified. This can happen only if the testers can test as soon as a story is coded.

Let's revisit our earlier iteration example. Once Story A is ready for testing on Day 3, we recommend that the tester sits with the programmer before the code is even checked in to get a quick demo. If together they find any issues, the programmer can fix them before declaring the story ready for testing. That's a short feedback cycle.

Continuous integration is another example of a short feedback loop (see more about this in the section on automation, below). Unit tests run as soon as code is checked in, so programmers know within a few minutes if a check-in caused a regression failure. Of course, each programmer can and should run the unit tests before checking in, but individual environments may not replicate all aspects of the regression test environment.

When the rest of the automation regression tests are run, they provide more feedback to the team. The shorter the cycle between runs, the faster any issues are found and can be fixed. The automation also frees time for testers to do exploratory testing.

Teams should use their iteration retrospectives to make small, incremental changes to their process. If testing is squeezed to the end, for instance, determine what might solve that problem. One rule might be to have the first story ready to test by Day 4 of the iteration. Another might be that no new features can be introduced the last day of the iteration. These adjustments will make the overall changes gradual and sustainable.

AUTOMATE

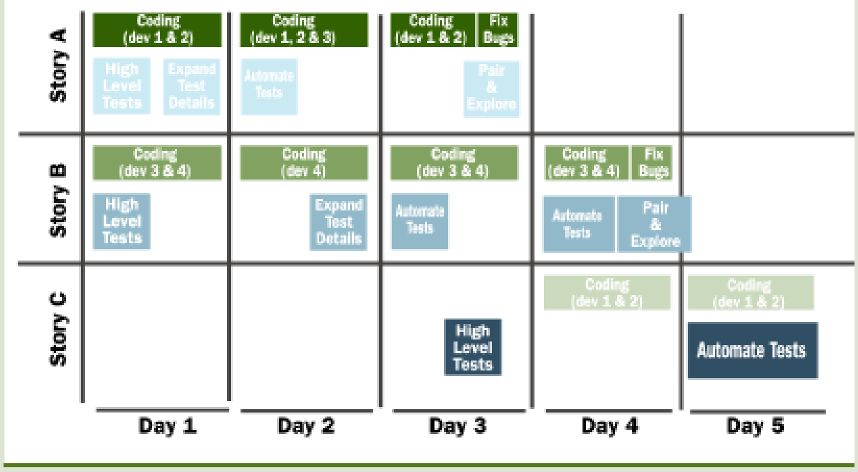
A key purpose of our tests is to guide the programmers to write code that ensures the product does what the customer needs it to do. An essential side effect of both TDD and ATDD is to produce automated regression tests that tell the team if a new code check-in breaks any existing functionality. Several aspects of automation — continuous integration, the automation test pyramid and automated regression tests — are critical to helping testers keep up with programmers during an iteration.

Continuous integration (CI), a build process that integrates all the code, compiles it, runs tests to verify there are no regression failures and deploys the latest code to testing sandboxes, is a mandatory component of any team's infrastructure. You can't get along without CI any more than you can opt out of using source code control. Without continuous integration and the quick feedback it provides, testing is sure to become a bottleneck.

For most teams, CI isn't hard to set up. There are many solid CI tools, both open source and commercial, and most development teams have the expertise to set up the build process. There are even tools such as TestifyWizard that speed up creating projects, tests and builds. (See sidebar on next page for links to CI tools).

Automated test pyramid: Cost is a critical aspect of return on investment

FIG. 1: TESTING TIMELINE



LINKS TO CI TOOLS

ThoughtWorks CI Feature Matrix:
http://confluence.public.thoughtworks.org/display/CC/CI+Feature+Matrix

TestifyWizard project startup wizard:
http://code.google.com/p/testifywizard/

Agile Alliance Functional Test Tools feature matrix:
http://bit.ly/AgileTestTools

Austin Workshop on Test Automation 2009 Automation Tools List:
http://awta.wikispaces.com/2009ToolsList

(ROI). Generally, lower-level tests such as unit tests cost less than longer, more complex tests to write and maintain, and the team can keep up with testing more easily if it automates as much testing as possible at the unit level. The next best choice from a cost perspective is testing under the GUI, at the object or presentation layer.

Today's GUI tools enable us to design relatively robust tests that aren't a maintenance burden, but GUI tests still run much more slowly than tests that don't have to navigate the UI. Experiment to find the minimum GUI test automation that will protect the software.

Regression tests should be a good thing — they're intended to ensure our testing "keeps up" with development, so testing doesn't devolve into a separate phase that lags behind coding. Yet we hear complaints: "We spend too much time maintaining our automated tests." "Our tests failed, but it's hard to figure out why."

Automated regression tests require the same care and feeding as production code. We must design our automated tests so that when they fail, the cause is obvious, because we need to be able to change automated tests relatively easily if the application changes — especially the UI, in the case of GUI tests. We also need to consider the tests' maintainability, because we need to identify issues and update the tests as part of our iteration tasks.

The purpose of automated regression tests is to provide quick feedback as to whether any new code check-ins broke existing functionality. To ensure speed, we must automate wisely, using risk analysis to determine what to include in our regression test suites. The right regression tests will identify bugs before they reach production, saving a lot of money and increasing ROI.

USE THE 'WHOLE TEAM' APPROACH

The team needs a shared vision of what "done" means for each story. All team members must realize they're responsible for quality, and that anyone can take on a testing task if necessary.

Quality in the product starts with the first discussion of a new feature, getting the assumptions out in the open and defining acceptance tests. We can't test quality into the code, but the team can "bake quality in" with good coding practices, such as TDD and ATDD. Using the "whole team" approach, the team collaborates, focuses on finishing one story or slice at a time, and marks each piece "done."

The team commitment to quality starts in planning meetings. Say, for example, there's a story for a batch process to parse, validate and upload employee census files. Tina Tester asks, "Can we write FitNesse tests for this story?" Programmer Paul replies, "Yes, let's slice up the story. The first slice is parsing, and we can write FitNesse tests for that. The second slice is validating, we can write FitNesse tests for that too." Tina wonders, "How will the batch job report errors and warnings from the validation?" The team and the customer discuss different ways of logging and reporting errors. Paul suggests a way to handle error reporting that would let them easily write a fixture to verify the error log files. They write and prioritize task cards accordingly.

[TEAMS SHOULD USE THEIR ITERATION RETROSPECTIVES TO MAKE SMALL, INCREMENTAL CHANGES TO THEIR PROCESS.]

Because the whole team takes responsibility for making sure all test activities are completed for each story before the end of the iteration, the entire team also participates in choosing test tools, making sure test environments are ready and the latest code can be deployed to them, finding ways to design testable code and planning time for essential testing tasks, such as exploratory testing. If a regression test fails — whether at the unit, functional or GUI level — the whole team is responsible for immediate attention to making it "green" again.

Perhaps there's testing the team can't do, because it doesn't have access to the right test environments or the necessary skill set. Some organizations may plan to do load and performance testing later in the release cycle rather than during each iteration. Some companies may require a security audit. No matter who ends up doing the actual testing or how it gets done, the whole team makes sure all these activities are planned and executed.

A daily standup meeting helps us remember we're working as one team to accomplish common goals. If we look at the storyboard on Day 6 of a 10-day sprint, for instance, and find several stories with many testing task cards in the "to-do" column, we know we have a problem. The team may decide to stop working on the lowest-priority story in progress and have a programmer pick up some testing tasks on the highest-priority unfinished story.

When the team puts good feedback tools, such as continuous integration, in place, it can ensure there's always a stable build to test each day, and a production-ready deliverable at the end of the iteration. The worst-case scenario is that perhaps the lowest-priority story wasn't completed, or a "nice-to-have" slice of a story must be carried over to the next iteration. Customer expectations are met, there are no leftover testing tasks to be squeezed into the next iteration and the team hasn't built up a queue of unfixed defects. The whole team is ready to start new stories, with no waste or rework.

NO PHASES, NO GATES!

Releasing business value frequently, while maintaining a high quality level, works best when we make sure quality is built in and testing isn't squeezed to the very end of an iteration. Don't let your agile team slip into a mini-waterfall, with short iterations divided into specification, design, coding and testing phases, or testing lagging an iteration behind coding. It's not sustainable over the long term, the team will start cutting corners to "save time," code won't be protected with automated tests and there will never be enough time for exploratory testing.

If you want the benefits of agile development, and the ability to release a solid product at least once a quarter, get your team working together. Prioritize those stories, slice them up small, and focus on finishing them one at a time. Your team will develop a rhythm of testing, coding, testing some more. Automate your regression tests and ensure quick feedback so the team stays on track. Collaborate with customers and each other to keep discovering ways to do your best work.

Don't Let Customers Catch Bugs You Miss

STOP Defect Leaks!

By Catherine Powell

Bugs come with many different stories. Some you find and fix so they never see the light of day. Others you decide are not significant enough to merit fixing — if a customer happens to find them after product release, you can address them then. Still other defects are never discovered by anyone — they lurk forever, wreaking no havoc. And then there are the bugs of which you remain blissfully unaware — until a customer alerts you to them.

This is the tale of that last category of bugs. This is the story of defect leakage.

DEFECT *WHAT?*

Defect leakage measures how many defects in the software were missed during testing but found after the product shipped.

You've probably seen the classic "costs of defect resolution" graph, which illustrates that the later it gets, the more expensive it is to fix a bug. Let's say your widgets have to register with your frobbles before the frobbles start running. Catch it in requirements? No big deal; just add it. Catch it in design? Not too bad; just update the REST API and workflow designs to make sure you don't have frobbles running amok or widgets that don't register. Catch it in development? A bit of refactoring, probably. Catch it in test? Shoot, now you have to grab a developer who's already moved on to the next project, and probably let the release date slip a bit. Catch it in the field? Oooh, at this point you have to find a way to patch all your customers, adjust your billing so as not to charge them for the frobbles that didn't have registered widgets, and then patch everybody else. You may have to push the next release, too, because this defect fix is bound to derail development for a while.

The point: Defects are expensive in terms of money, time and reputation, so any bug a customer finds hurts you. That's why you want to get your number of defect leaks as low as possible (zero would be good!).

HOW TO MEASURE DEFECT LEAKAGE

For a defect to be counted as "leaked," it has to meet only two criteria:

1. The engineers and testers have to have missed it
2. A customer has to have found it down the line.

Seriously, it's that simple.

As with many measurements, a number alone doesn't really mean much. Let's say your number of leaked defects in a given release is 500. If you're, say, MassiveCo shipping a major OS, that's probably not too bad. But if you have one customer and the product is two

Catherine Powell *(blog.abakas.com) writes about what she knows: teams, testing, and the everyday adventures of shipping software.*

text fields and a single output, 500 leaked defects is terrible. There has to be a comparator to provide perspective on the number of leaked defects. That comparator is the total number of bugs found in a given release by everyone involved — the testers, developers and customers.

The defect leakage rate is simply the number of bugs leaked divided by the total number of bugs. For example, if you found 100 bugs in a release, and in the field you found 10 leaked bugs, your leakage rate would be 10 percent.

A bonus of sorts for high-functioning organizations is the ability to track defect leakage separately for each bug priority level. You wind up with a table like that in the example in Table 1. Comparing your defect leakage rates across priority can help you determine which test changes to make first.

WHAT'S A REASONABLE DEFECT LEAKAGE RATE?

Zero is ideal, but 1 percent to 5 percent leakage is a more realistic goal in most cases. To determine an achievable rate, determine your current rate, then reduce that rate slightly with every release. At

Illustration by The Design Dvia

TABLE 1: SETTING PRIORITIES BASED ON DEFECT LEAKAGE RATES

Priority	Leaked Defects	Total Defects	Rate
P5	2	75	2.7%
P4	1	36	2.8%
P3	5	163	3.1%

some point, reducing that rate will become more expensive than simply dealing with the leaked defects — that's when you stop. (This is the way most efforts work, from testing to performance improvements. Keep going until it's more expensive than stopping and dealing with the consequences.) How much you try to reduce your defect leakage rate depends on where you start and your other priorities.

It's not easy to reduce defect leakage, but once you measure it, you're well on your way. There are only a few things you need to know about each leaked defect:

- Affected customer(s)
- Type of defect
- Version in which it occurred

You don't want to dive too deeply into details; you're trying to see a pattern. Specifically, you're looking for: (1) releases in which you leaked a lot of defects, (2) patterns of components or test types in which you're missing defects, or (3) customer environments in which defects consistently occur. This will help identify the areas in which you need to improve — not just in testing, but possibly in development practices, deployment guidelines or even the way you set requirements.

Pick a specific area and take a closer look. What's causing a particular customer's problems? Is it an interaction with something in the customer's environment? Some use case you haven't even thought about? A particular part of your product? Once you identify the source of the problem, you can determine how to improve it. Multiteam solutions are likely — this isn't something any one individual or group can go off and do alone. Defect leakage often points to a systemic problem. Remember, if there were a simple fix, you'd already be doing it!

[You don't want to dive too deeply into details; you're trying to see a pattern.]

Consider these examples:

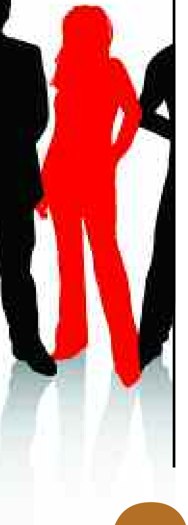
The Problem Customer: Sometimes you find that a disproportionate number of your leaked defects come from one customer, or one kind of customer. Keep in mind these are real defects — it's not that the customer is complaining about nothing. Evaluate the issues and figure out what these customers have in common. Maybe it's an interaction with a third-party product. Time to get that third-party product in house and start using it. Still missing defects? Work more closely with the clients to figure out what you're doing differently — it's a good bet they want to help make sure these problems are solved, pronto.

The Too-Big-for-Our-Britches Customer: Some problems are caused by clients attempting to use systems at higher volumes than intended. Even if a certain usage level is not ideal (who'd put a million files in a single folder?), it's still causing problems. So you need to design for and test that specific use case. Check your designs and make sure your UI has pagination or other metaphors to handle large amounts of data. Make sure your architecture can handle the growth. And follow through by testing with large amounts of data — don't make any assumptions.

The Sensitive Component: When you study the issues closely, you may discover a component that just isn't working in the field. Most of your problems are this thing's fault. Fortunately, this is a relatively straightforward engineering problem. Consider a rewrite if things are truly bad. If they're not so drastic, refactor, add code reviews, and institute more formal component mentorship and oversight. Get it stable; your customers will thank you.

Voices of Women Testers

6 Industry Pros Share Their Unique Perspectives on How to Survive – and Thrive – In Today's Challenging Industry



9 Tips to Encourage Collaborative Testing

By Lanette Creamer

Collaboration among software testers effectively brings another perspective to early software evaluation and helps create an environment ideal for testing multi-user scenarios, permissions, security, custom configurations, integration, and cross-platform and cross-product workflows. In addition to providing additional opportunities to uncover bugs, it provides a chance to share product knowledge and testing techniques.

Given these advantages, why isn't more collaborative testing being done? Because most testers don't fully understand the value of collaboration, and most companies don't know how to entice them to participate. Only by spelling out the benefits for testers and clarifying the value you place on their time and talent can you draw them in. Here are nine ways to make it happen, with real-world examples based largely on my experience at Adobe.

Photograph by Blue-Fox

1. ENLIST 'VOLUNTEERS'

Like most quality engineers, Adobe's QEs sometimes struggle to identify testing gaps between teams. But when a bug impacting a key customer was identified between products several years ago, Adobe's QE managers teamed up to investigate the situation and discovered they could address other, broader testing gaps by working together on certain projects. After resolving the initial dilemma, they began identifying larger gaps and championing the concept of collaborative events, encouraging both executives and testers to support the idea.

Collaborative projects do take testers' time and focus off their regular, single-product testing, so when you request participants for your collaborative events, limit their involvement (we keep it to one or two days maximum per tester), and get managers' approval before letting anyone sign on.

All that said, the combination of enthusiastic participants with the appropriate skill set, dedicated hard work and fast sharing of data generally proves to be a plus for everyone involved, including the company as a whole. Through collaborative test events at Adobe, for example, we've found bugs that have been fixed before every Creative Suite product demo to date, and we've identified an incompatibility with Flash Player versions that would have impacted most Creative Suite users.

The return in savings far exceeds the investment, due to the severity and scope of the bugs found that wouldn't be identified without collaborative testing. It's virtually impossible to quantify cost or return on investment because we find these bugs before shipping, but to put it in perspective, note that the cost of reshipping even the smallest of the Creative Suites in just one language on one platform is many millions of dollars, so if we find even one bug and prevent the reshipment of one Suite, we save potentially millions of dollars per release.

2. CREATE CELEBRITIES

Don't dream of saving the day, create the day when one tester whispers in the ear of another, "Did you hear what happened last night during the Bug Bash? Chris brought the server down again!" Testing with others makes the best bugs the stuff of legend, and the testers who uncover them become stars. The search for software defects becomes a delight, and the spread of success stories points the test team naturally toward the bugs' nests.

3. STIMULATE COMPETITION

Who's buying the drinks? Make a deal: If no new bugs are found as a result of the fix that went in, the QE team buys rounds for the development team. If the QE team can find a bug severe enough to force a fix, drinks are on the developers!

Before we shipped Adobe Creative Suite 2, we'd taken some bug fixes for the installer. Everyone got involved, all the way up to the director of testing and down to the newest intern.

But think about it: As a tester, do you really want your manager, who hasn't been in the trenches for years, to find more bugs than you do? How about the developer who wrote the feature? Sure, the developer might be great at reviewing his or her own code, but you want to shine when time is ticking down to the final seconds.

The competition can get intense as the voting for the most catastrophic crasher, longest lag or scariest security slip is under way. The key for competition to work well is to keep it exciting and positive, and make product quality the real winner.

4. FOOD, FUN, FRIENDS

There's more to collaboration than the end result — there's the camaraderie that develops throughout the process. When most testers walk past a lab and see pizza boxes piled up and people gathered around a computer trying to solve a problem, they can't resist belting out some suggestions: "Try entering the umlauts!" "Try a blank filename!"

Teambuilding is a complex topic to be studied in graduate management courses based on thick textbooks filled with organizational theory…or not. Sometimes it's as simple as, "Hey, Ajay, is this supposed to work with Safari?" Demonstrate this kind of spontaneity and set an example for the rest of the gang.

5. PROMOTE YOUR PROJECTS

If the project you're working on involves the use of some exciting new technology or tools, draw attention in creative ways to stimulate participation. To promote collaborative testing on Adobe's Creative Suite 3, we designed *film noir* movie posters — just seeing the image of the engineering manager in a fedora was enough to catch the eye of potential volunteers. Less politically astute but perhaps more memorable was our bearded test lead in a bikini and go-go boots, part of a promotion for an InDesign 2.0 project. The QE manager probably burned these images by now, but they served the purpose at the time. Fun is underrated, and while you may wonder how to strike the right balance, it's still legal!

[The key for competition to work well is to keep it exciting and positive, and make product quality the real winner.]

6. GET MANAGEMENT BUY-IN

The difference between "What kind of testing is *that*?" and "Let's do that user experience testing every week!" can be as simple as a few meetings with the QE and engineering managers, including demos and discussion of bugs that have been found during collaborative testing.

Wrapup e-mails from the event leader sharing results of the activities and highlighting the bugs found during the test event also encourage management buy-in for future collaborations. Here's an example of an actual entry from one such e-mail:

Photoshop: Bug 554323 — Gradients applied in and Illustrator PDF drop out when opening in Photoshop. Status — Fixed in build 214.

7. SHOW YOUR APPRECIATION

Our Photoshop 5.0 Bug Hunt was a fast-paced dual-location event (we held it simultaneously in Seattle and San Jose) that generated a lot of buzz thanks to the prizes we offered the winners — no big bonuses or intercontinental vacations, but food and gift cards in varying amounts to add to the excitement and show our thanks. Developers from teams throughout our offices came into the lab and grabbed a slice of pizza and a list of new product features and changes so they could try their luck at finding bugs. It was fun, it brought people together and it paid off — literally — for some.

8. TRAIN YOUR COLLABORATIVE TEAM

A test center may not look like a classroom, but in a sense that's just what it is. Every tester needs some basic training specific to the company's software and related products. Testers new to Adobe, for instance, spend a day getting to know how the Creative Suite installer works, how to use the restore function in Mac OS X, key Photoshop and Illustrator upgrade features, even some ActionScript 3.0 and Cascading Style Sheets. We also provide new collaborators with a list of contacts on all relevant product teams — this often proves to be the most useful training tool of all!

9. SHARE YOUR TIME AND RESOURCES

When testers offer you their time and expertise, reciprocate. At the time of the collaboration, include their ideas and areas of concern, or set aside a different time if you can't test everything at once. Being a good leader is often touted as an important tester skill, but being a fantastic follower is just as vital.

What's more, if a tester asks you to discuss some test-related ideas over lunch or coffee, try to make time to go, listen and be present. Be a person, not a cog in the machine. Machines all do their own jobs, never improving for their proximity to greatness. Even if you're in proximity to staggering mediocrity, you can learn something of value to apply to a future collaboration. ☒

Lanette Creamer *(lanette.creamer@gmail.com) is quality lead for Adobe Systems, where she coordinates cross-product testing events for the company's Creative Suites. She has 10 years of software industry experience.*

The Politics of Testing: Making Conflict Count

By Elisabeth Hendrickson

I knew that the VP facing me across the table would not want to hear the news I was bringing him. "There's a problem," I began. Immediately his expression grew stony. I explained what we'd found in testing and why it was a problem. He listened silently, jaw clenched, face reddening. When I stopped talking, he took a deep breath and glared at me. My stomach fell.

Then he launched a full-fledged attack. "Why is your team just finding this now?" he demanded. "What have you people been doing all this time?" His anger and frustration were palpable. And he was just getting started. He went on to accuse me of incompetence and my team of laziness.

That conversation was years ago, but just thinking about it still dredges up feelings of helplessness, defensiveness and bitterness. That day I realized that the most difficult part of testing isn't the actual testing; it's the politics. It's dealing with the fallout when tests uncover information no one wants to hear.

HAVE COURAGE

As uncomfortable as it was to bring bad news to the VP, it was my job. As a tester, it's my obligation to tell the truth, and that's what I'm paid to do. Sugar coating the test results might make them more palatable to stakeholders who would rather cling to the illusion that everything is fine, or that everything *would* be fine if it weren't for those pesky test results. But that won't change reality.

Still, in the face of such intense emotions, it's sometimes hard to keep that in mind. Dealing with these situations requires a certain amount of intestinal fortitude — you have to stand your ground and speak the truth even if you're afraid your stakeholders would prefer to shoot the messenger rather than deal with a disappointing message. I find comfort in Jerry Weinberg's saying, "It's not a crisis; it's the end of an illusion."

Having had many such difficult conversations, I've developed a strategy for handling poor reactions to bad news. I don't allow myself to get bogged down in accusations or blame. Instead, I respond to the essential concerns behind the anger and frustration, and seek ways to work with my stakeholders to address those concerns.

Let's take an example. Imagine this VP, fists clenched, face red, launching into a diatribe at the news I've just delivered. As family therapist Virginia Satir said, "The problem isn't the problem; the coping is the problem." At this point my problem isn't the test results; it's my VP's reaction to the test results.

[The most difficult part of testing isn't the testing; it's the politics. It's dealing with the fallout when tests uncover information no one wants to hear.]

Sure, I could hook into the implication that my test team and I have done nothing for months. I could leap to defend myself and my team. I could attempt to shift his anger onto the programmers and ask how all those bugs got into the code in the first place.

I could. But I won't.

Attacking or defending would only escalate the tension, and wouldn't make the situation any better.

Instead, I respond by reflecting the essence of the VP's underlying concern:

"It sounds like you're frustrated that we're still finding bugs and that the schedule is slipping as a result. I'm frustrated too. Would you like to talk about what we're doing now to address the situation?"

This response serves two purposes:

- It acknowledges the VP's very real and perfectly valid concerns. It signals that I'm listening. Just sympathizing is usually enough to diffuse the worst of the explosive attacks.
- It provides a chance to shift the focus of the conversation to a more productive "what can we do to move forward." After all, I can't change the past. But together we can change the future.

HOLD YOUR GROUND

At this point, the VP has a choice. He might attack again. If he feels he's been barraged by his stakeholders for the schedule slips, for instance, it would be a

Photograph by Endostock

natural reaction to pass that pain along:

"Yes, I'm frustrated! There's no reason you should have found these issues so late! Why weren't you testing earlier?"

If he attacks again, I'll continue reflecting back the valid concerns I hear him express, and offer to work on those concerns:

"I understand your frustration that we're finding these issues late, and I agree this is a problem. Would you like to talk about what it would take to identify problems earlier?"

Eventually the VP will either agree to have a productive discussion with me or he'll get tired of my approach. If he gets fed up, he might yell at me, he might just stop talking and go back to glaring at me, or he might end the meeting altogether.

That's OK. He's entitled to be angry, and he's entitled to express that anger. But I don't have to be the target of his fury. If any of these things happen, I'll leave the meeting with a parting invitation:

"I'd like to work with you to make sure we don't end up in this situation again. I can see this is not the best time to have that discussion, so let's try again later. Let me know when is good for you."

It's more likely, however, that after a few rounds of the blame game, the VP will shift his focus from attacking to trying to solve the problem:

"Fine. Tell me what you think it would take to find these problems earlier."

Now I have his full attention, and a fantastic opportunity to open a conversation about what I think needs to change to improve our test outcomes.

FOCUS ON THE FUTURE

If I haven't prepared, that opportunity will be wasted. So before I even walk into this meeting, I will have spent some time investigating the situation to understand what went wrong and to form my own opinions about what it will take to improve things. That way, I can make concrete observations and recommendations.

I might say something like:

"At the moment, all our regression tests have to be executed manually. That takes days, and it means the feedback cycles are too long. I'd like to start working on getting automated regression tests in place."

Or I might say:

"The way the teams work now, we have a QA group working in isolation. I'd like to integrate the QA effort better with the development effort."

Whatever the root cause and whatever solution I plan to recommend, I show up ready to open the discussion, talk about what it will take to implement my suggestion and aim for a positive outcome to a grim situation.

This approach isn't easy. It requires a little courage, a lot of patience and a stubborn refusal to get caught up in accusations. But the end result is worth it. Out of the worst, most uncomfortable conflicts comes a genuine opening for productive conversation. ⊠

Elisabeth Hendrickson *(http://www.qualitytree.com/) is the founder and president of Quality Tree Software, a consulting and training company dedicated to helping software teams deliver working solutions consistently and sustainably.*

The Power of Pessimism

By Sharon Robson

No doubt you've heard of Murphy's Laws, but did you know that Murphy was a software tester? In *"Testing Extreme Programming"* (Addison-Wesley, 2002), Lisa Crispin and Tip House identify Murphy as a tester, which suggests he was lifecycle-agnostic — indeed, his principles can be applied to all the main factors testers consider when defining their approach to almost every project: people, process and product (software/technology).

Of course, while software testers are professional pessimists, they have an underlying foundation of optimism — they're "positive" something will go wrong!

So Murphy's Laws can serve as an excellent checklist to help focus your testing. Not only can they guide you to development of a more thorough testing approach, but they also offer valuable insights into achieving better interaction with team members and stakeholders throughout the software development lifecycle.

Here, then, is a list of 10 of Murphy's Laws as they apply to software testing:

1. Anything that can go wrong will go wrong. Clearly, Murphy was thinking about risk and risk analysis here. This is a great people and process factor. Bear it in mind when you think about your system and your test effort: Identify what might go wrong in your approach and make provisions to manage the risk. You can also extend this rule to the defects to look for in the software/technology aspects of a system — look for what went wrong previously. This will indicate where to focus some of your testing effort.

2. You'll always find something in the last place you look. When it comes to software/technology, this is invariably true. Defects are usually well hidden, not obvious, so you'll have to think hard about where they might be and carefully structure your testing to find them. Testers need to know about the systems they're working with and the processes the systems are enhancing to make sure they're being implemented correctly. Testers also must consider how the end users are going to use the system and make sure the actual user scenarios are tested prior to release, as part of the process, to make sure these defects are not found last.

3. Anything you try to fix will take longer and cost more than you thought. From a software/technology and process point of view, no defect is a "quick fix," no matter what the developers tell you. Not only do you have to make sure the defect has been fixed, but you have to think about regression — and test for it.

4. If you fool around with a thing long enough, you'll screw it up. Testers know this one by heart — it's our mantra. An additional consideration is the application of time to our test effort, to ensure that the system's nonfunctional attributes are tested as well as its functionality. This is definitely a software/technology factor, as the nonfunctional attributes have to be designed into the system from the beginning, taking into account the product's required "uptime" and other quality attributes. Reliability testing is a key area often overlooked in the test effort.

5. If there's a possibility of several things going wrong, the one that will cause the most damage will be the one to go wrong. When it comes to testing, this is a plus – as long as we find that worst possible problem before the system goes live! This is a process factor; we need to identify these potential defects early in the

Photograph by George Martinez

lifecycle and test for them as soon as we can, using both static and dynamic techniques if possible.

6. When you demonstrate a broken appliance for the repairperson, it will work perfectly. As every tester knows, a typical developer response is, "It works on my machine." As part of the tester's approach, and a people factor, make sure you know how to reproduce each defect raised, clearly and concisely, and be able to articulate the data and approach used. Also, make sure the environment and the test were set up correctly.

7. Build a system even a fool can use, and only a fool will use it. An oft-overlooked people factor, this is all about usability. Know who'll be using the system you're testing. Make sure to try all the appropriate business scenarios when completing your acceptance testing. Be sure the system meets everyone's expectations. Don't focus only on the people who asked for the change — consider all user levels and classes.

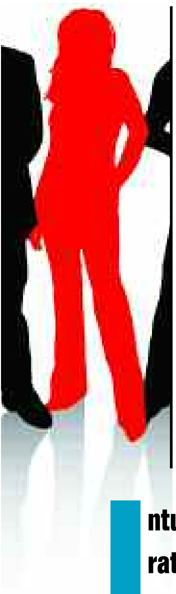
8. There's never time to do it right, but there's always time to do it over. This is so true in terms of testing — there are always overruns in delivery to the test team, but there's always enough time to redo the work when the delivered build is incorrect. This is a process factor within the development lifecycle, so we must consider how long it takes to resolve any issues or defects. Remember to include time for these "do-overs" in your time estimates.

[Many a defect can be an unintended side effect of the project approach.]

9. Nature always sides with the hidden flaw. No matter how well hidden defects are during testing, and no matter what obscure piece of code they're buried in, users will find them eventually. As part of the process, make sure when doing a risk analysis that you consider *all* risks, no matter how small or seemingly irrelevant. From a software/technology point of view, it's also important for testers to be knowledgeable about the tools and techniques used to build the system under test; many a defect can be an unintended side effect of the project approach. Keep an eye out!

10. Never argue with a fool; people might not know the difference. This is a pure people factor. Relationships with developers are extremely important, so don't assume the developers of the product you're testing aren't as invested in quality as you and the rest of the test team. Remember that you too have to be clear and articulate in conveying your message. Focus on listening and learning as part of your communication plan. A tester's credibility is vital to the information he or she delivers, so make sure you have a solid understanding of the system you're testing, and don't risk casting yourself in the role of the "fool." ⊠

Sharon Robson *(SharonR@softed.com), BSC Hons, Grad Dip IT, CTAL-AT, CTAL-TM, is a trainer and consultant specializing in software testing for Software Education.*



Putting Intuition to the Test

By Nancy Kelln

Intuition: "The act or faculty of knowing or sensing without the use of rational processes; a perceptive insight; a sense of something not evident or deducible; an impression."

—www.thefreedictionary.com

Like most females, I was born with "women's intuition." I use this intuition as one of my own personal testing tools, not only during test execution, but also when dealing with my project team and stakeholders, planning testing activities and adapting test plans and approaches as projects change. It doesn't replace "scientific" tools like requirements-based testing and other techniques and metrics, but it's a reliable addition.

As a tester, I need to determine if the part of the application I'm evaluating feels solid. I rely on my intuition to tell me, "Yup, this piece looks good. I believe I've tested this area sufficiently to be comfortable with it and I'm ready to move on to the next part of the test."

Does this feeling correlate to the number of test cases I've run? Maybe, but I usually do exploratory testing, and my exploratory test measures aren't always easy to capture, at least not in traditional test case reporting form — number of tests planned, number of tests passed and so on. Does this feeling correlate to the number of defects I've found? Maybe, but many times I've felt uncomfortable with an area under test; I haven't found significant defects to support the feeling. I just knew my gut was telling me something wasn't right.

We use our intuition every day when crossing a busy street at an uncontrolled intersection. When there is traffic coming we judge how fast a car is moving and weigh that against how long it will take us to cross the street. We don't rely on metrics to tell us the exact speed the car is traveling in miles per hour or how long it will take us to cross in seconds or minutes. Instead, we take our best guess at these facts and go with our gut, even in potentially life-threatening situations. As a tester, I don't always have all the facts about the application under test, so I've learned to trust my hunches in all aspects of my work. I believe that intuition is a unique tool all testers — not just women — should be encouraged to use and, indeed, celebrate. By leveraging their intuition when they perform the various aspects of their jobs on software development teams, testers gain new insight into the testing, allowing them to test the product in ways that preplanned, scripted testing might not have addressed.

INTUITION AT WORK

As testers, we work with developers, project managers, business users and a variety of other stakeholders. Building and maintaining these professional relationships can be complex. I find that testers who use their intuition in these relationships are more successful in developing strong relationships with the people on their teams.

This comes in particularly handy because testers are often the bearers of bad news. We tell developers about problems in their code. We alert project managers to issues that could impact their timelines. We inform business stakeholders about missed requirements or ways in which the system acts differently from expected. Because no one likes to hear bad news, understanding how people will react and gauging when to communicate is an important skill for testers. Being an astute observer of mood and body language helps me decide when and how to deliver my message so it comes across clearly and conveys the appropriate level of impact.

Another area in which intuition pays off is in determining test plans. I still gather all the information required of a risk-based approach, but I let the information I've gathered through discussion with my team lead to a gut feeling that helps guide my test approach and determine the extent to which I test each area of the application.

My intuition probably plays its most critical role during test execution, especially during exploratory testing. With most applications, there's no way to manually test every scenario or condition. Therefore, we need some method to determine when we've completed enough testing to feel confident in the area under test. Many tools and processes have been developed to try to prove appropriate coverage has been performed — requirements-based testing is a good example. However, I've seen many testers use the art of their intuition along with the science of testing metrics to successfully gauge when they've tested enough and have achieved adequate coverage.

Finally, I use my intuition to adapt my test plan to the always changing demands of software testing. One reality of software development is that often we don't have all the facts, especially in the early stages of planning. As we move along our project timeline, we uncover new information and must adapt accordingly. For instance, I may begin to feel more or less confident about areas I've already tested or areas I still plan to test based on facts that emerge and my instincts about them. When this happens, I adjust my plan accordingly.

CONTEXT COUNTS

Although intuition can be a strong tool, it must be adapted and applied in context. For example, I've found that intuition doesn't scale well to larger projects. Depending on the size and scope of the project in which I'm engaged, I may not have extensive awareness of the project goals, objectives, timelines and risks. I still leverage my intuition, but I remain sensitive to the importance of seeking out additional information before letting my gut feelings guide me. In these cases, my intuition tells me to base my decisions more on facts and less on intuition.

Because I've found my intuition such a valuable tool in my testing, I encourage other testers to rely on their intuition as well. And, by the way, I meant what I said earlier — women aren't the only ones who possess intuition. I've met some pretty intuitive male testers, too.

The bottom line: Not all software testers, regardless of gender, use their intuition, but we need to learn to trust our instincts when it comes to testing. We have to go with that gut feeling about what, when and how to test. And we have to report not only the metrics but the feelings about our test findings. Only then will we provide the complete picture of product quality. ☒

Nancy Kelln (nancy.kelln@shaw.ca) is an independent consultant with 12 years of diverse experience within the IT industry.

Illustration by Frank Boston

Use the Social Web to Build Your Career Brand

By Rosie Sherry

There seems to be no way to get away from the idea of the social Web. Twitter this. Facebook that. LinkedIn here. YouTube there. Everyone's talking about it, but how can we use it to build a personal brand and benefit the wider software testing community? My own experience serves as a good example. In the running up to becoming a mother I knew my life — both personal and professional — was about to change. But it wasn't until I actually had my first child that I realized just how different things would be. I suddenly found myself more restricted careerwise than I'd anticipated because of the personal choices I'd made.

So my next decision was to do whatever I could, using the tools and capabilities available. This meant connecting with other testing professionals online more often than face to face and establishing myself as a freelance testing consultant.

Toward that end, one of my first steps was to set some goals for myself: get my name known and find local, flexible or remote testing work, preferably involving Web applications and accessibility, my primary areas of interest. From that point on, I geared everything I wrote and communicated, online and off, toward meeting these goals.

I found that this approach suited me, and my clients and family, all quite well. In fact, I discovered that the social Web is ideal for achieving visibility, and building a brand and a business, across the globe. It also allows you the flexibility to adapt as customer and personal needs evolve.

With that in mind, here's a look at some tools to help build your personal brand.

BLOGGING BENEFITS

First, let's get this straight: Blogging is not for everyone. Every successful blog is unique, with content that reflects the writer's personal or professional goals. For me, it's about reaching out to others, sharing knowledge, being helpful and interesting.

But there's good reason there are relatively few software testing blogs out there. Being interesting and helpful isn't always easy, and finding the time to keep it up is hard. Ideas can dry up quickly. It's easy to develop writer's block. And an empty or boring blog won't do you any favors.

However, there are clear benefits: Writing regularly sharpens your communication skills. It provides a point of inspiration and feedback for developing ideas in cooperation with other testers, local and worldwide. It allows you to develop conversation, community, even friendly controversy, one on one or in groups, letting you and your potential clients develop a pressure-free rapport. And it's a terrific confidence booster.

[Blogging is not for everyone. But there are clear benefits. It provides a point of inspiration and feedback.]

My top tips for blogging are:

- Get a domain and stick with it for the long term — it's great for branding and search engine optimization.
- Keep your entries original, useful and intriguing.
- Let your passion and character emerge in your writing and your design.
- Be professional and respectful at all times; search engines are like elephants — they never forget.
- Include a short, readily accessible description of yourself.
- Make your site easy to navigate, and make it simple for people to contact you.
- Participate in other blogs and online conversations — don't be afraid to reach out to other tech professionals (most of them don't bite!).

When do you know your blog is becoming a success? Simple measurements include increases in traffic, quantity and content of visitor feedback and, in some cases, a great job lead or a rise in the number of inquiries you get for testing-related consulting assignments or speaking engagements.

TWEETING FOR TESTERS

The booming microblogging service Twitter is ideal for those who've never gotten around to blogging but want to be their natural social selves online.

I've found that the principles of blogging generally apply to "tweeting," too, but Twitter tends to focus more on the social than the professional. To build your business using Twitter, the best bet is to dive in and participate. Start by searching for other testers in online Twitter directories, find out whom others in the testing industry follow, and check out Twitter lists others have created (you can find my list of testers at http://twitter.com/rosiesherry/softwaretesting).

On Twitter, you can view updates from everyone you follow in one stream. So, for instance, you can follow numerous people in the world of software testing, getting a single stream of tweets about testing when you want them, and jumping into relevant conversations at your convenience.

My top tips for using Twitter:

- Find interesting people to "follow."
- Participate often in conversations.
- Be yourself — transparency and honesty are crucial at all times.
- Make it easy for others to get to know you — post a short bio, and link to your Web site, blog or LinkedIn page.
- Help other testers and ask them for help as well.

By the way, if tweeting doesn't seem like your thing, you can still get some of the juice just by seeing what others are tweeting about. Start by checking Twitter search (http://search.twitter.com) for testing terms or follow custom Twitter lists (as mentioned above) for tips. Much of the content you'll find on Twitter can't be found through the more traditional search engines, so you may be pleasantly surprised.

Photograph by Photosani

JOIN ONLINE COMMUNITIES

There are many places online to participate too. Online communities include the likes of STP Collaborative (home base of this magazine) and the Software Testing Club (my "baby"), as well as LinkedIn and Yahoo groups. They all vary in their offerings — some serve primarily as discussion centers while others are mainly information resources — but they all have impressive members among their ranks, and you can sign on with minimal if any commitment, participating to share knowledge and gain exposure among peers at pretty much any time.

My top tips for participating in online communities:

- Be helpful and respectable.
- Promote yourself "naturally" — keep in mind that online communities aren't a place to blatantly peddle your skills.
- Maintain a consistent profile among all your online spaces (your blog, LinkedIn, Twitter and Yahoo pages, for example).
- Try to get to know some of the community members personally, as long as there's a valid reason (don't send en e-mail just to say hello).

It's easy to get carried away and waste valuable time (yours and others') because of the sheer number and size of online forums and communities. Choose a handful where you can focus your participation. They don't even have to be specific to software testing — I'm a big fan of crossing over into other niche communities (e.g., agile, accessibility, Web development). In fact, it would be great if more testers participated in communities devoted to other technology areas, to learn from our peers in related fields, widen our reach and spread the word that we really do exist!

GET FACE TIME

Online interaction is fantastic, but it's not a substitute for face-to-face interaction.

Many software testing events take place across the globe every year. Some are more formal, costly and time-consuming; others are informal, local meetups, often spurred by the growth of the social Web. Use the Web to set the stage, but attend as many of these meetings as you feel you can afford, both time- and moneywise. I've had some 10-minute "real-life" conversations with other testers that have proven valuable far beyond any amount of Web-based discussion.

My top tips for live events:

- Tear yourself away from the computer and go to them!
- You don't need to meet everyone, but try to be social.
- Put faces to names and introduce yourself to those who read your blog or follow you on Twitter, and whose postings you read as well.
- Attend other tech-related events, not just testing-focused gatherings
- Bring a business card or other handout that includes your contact details (my favorites are Moo cards, www.moo.com, but the choices are virtually endless).
- No local meetings? Start your own!

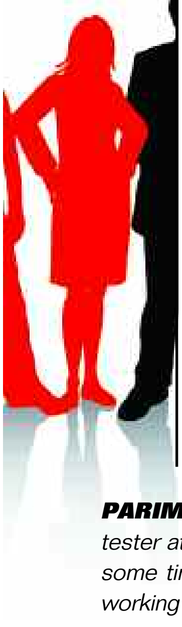
BE DIRECT

Testing consultant James Bach posted a blog entry in June called "Have Internet, Will Test" (www.satisfice.com/blog/archives/322) that raised an interesting point: Too often people are shy or scared of coming across online as overly self-promotional. They expect that just because they're online or have written an article, the offers will come flying in. But it doesn't work that way.

It's important to let people know what you want (or don't want). How can they help you if they don't know what you're looking for? It may be a work-related opportunity. Or perhaps you have a great idea for an article. Or you have some time to volunteer your worthy skills. The strong online presence you've developed will increase your chances of getting what you need — if you ask. Humans tend to be helpful, but they need to feel confident that they're supporting the right person for the job.

In effect, this means all the work you've put into building your personal brand and developing your professional reputation will help others decide if they're comfortable recommending you. Make that decision a no-brainer for them. ⊠

__Rosie Sherry__ (rosie@schux.com) is a U.K.-based software tester turned social media professional. Starting as a tester primarily for Web-based projects, she now supports the testing industry through the Software Testing Club (www.softwaretestingclub.com).



India's Women Testers Gain Visibility: A Conversation With Consona's Parimala Shankaraiah

***PARIMALA SHANKARAIAH** is a senior tester at Consona in Bangalore. We took some time to ask for her perspective on working as a tester in India today.*

STP: There are many great women testers in India; whom do you consider some of the most notable?
PARIMALA SHANKARAIAH: India is home to women like Dr. Meeta Prakash, Minal Deshpande, Anuradha Biswas and Rashma Samani, all of whom shine in the testing business. While Meeta defended a thesis in testability to win her PhD in software testing (the first Indian to achieve it), Minal Deshpande picked testing as her area of interest early in her career and is now head of testing services at Deloitte. Anuradha Biswas rose from being a quality assurance and testing associate to head of testing services at Infosys. Rashma Samani manages one of the largest groups of testers at NDS and is considered a very successful test manager. Likely many women I don't know about would be eligible for this list.

These are examples of Indian women testers who had great potential and cut through all barriers to establish their credibility. Outside the testing context, we have women like Padmasree Warrior, CTO of Cisco, who proved to the world that Indian women in the technology space are as competent as the men and women of any other country. All have inspired both men and women in the IT and testing fields. But why so few names? Shouldn't I have a huge list?

What are some of the challenges facing women testers in particular?
India was one of the first few developing countries to rid itself of the dogma that women should not go out to work. So that isn't our challenge.

Today, many Indian women are self-dependent with skills and work outside the home. Their contribution to the country's growth has been very significant.

Unfortunately, that doesn't mean they have escaped all problems. Women still have additional responsibility compared with men, taking care of their homes as well as managing office work. Most men in India enjoy this luxury, so their focus on work can be 100 percent, whereas women have a 50-50 split. Often, they manage to be as productive as men. But when they can't take time to interact with the testing community, it limits visibility of how good they are at testing.

What's your advice to women trying to do it all?
Women testers in India need to balance work at home and office. It's challenging to find time to interact with other testers, read testing books and articles, practice testing, blog about testing experiences…. Yet it's encouraging to see many women getting actively involved in these activities.

Patrilineal family structure is the basic unit of society in India. If any woman in the family chooses to work instead of being a housewife, the entire family decides whether to let the woman work or not. Fortunately, this is changing for the good. Many families today are more open to women going to work and helping women balance work and life. This increasing support makes women testers all the more enthusiastic to explore. The traditional pattern of women running around both home and office is changing. This is very encouraging for many women testers.

Do you see things progressing for women testers in India?
Yes, a lot! Indian women testers are slowly getting better at finding the perfect work-life balance. They have taken a giant leap from being average testers with 9 to 5 jobs to people with ambitious and accomplished careers in software testing.

A growing number are coming out in the open and talking about their experiences and learning with the outside world through blogs, in online forums, by attending and speaking at testing conferences and universities, and so on. Thanks to specialization, women testers are a lot more capable today in their testing skills, technical/scripting skills, and communication and other soft skills.

What forums are available to facilitate communication?
Many informal testing associations involving women testers have started in local communities to discuss testing challenges and how to overcome them. Such groups also reach out to and train women testers who are finding problems in learning new technologies and scripting languages or testing different types of products. These can be ideal groups for women mentoring women, where group learning is encouraged.

Weekend Testers (http://weekendtesting.com/) is one such peer testing group initiative. So far, more than 25 women testers have tested open source projects and learned to be better exploratory testers through Weekend Testers. Based on my experiences in Weekend Testers, I'm thinking of starting an all-women testers peer group that will discuss and address women-centric issues in testing.

Women testers are taking testing careers much more seriously than in the past. Using social media, women testers are getting connected with testers in different parts of the world to learn and share knowledge as part of a single global testing community. Through this, we see the passion and enthusiasm these women have exercised in passing through all the challenges and achieving success.

A few years from now, I hope to have another conversation with you about Indian women testers, and I hope the list of globally recognized testers has grown substantially by that time. ☒

A Wealth of Resources For Women Testers

IF YOU OR YOUR WOMEN CO-workers, managers or staff want to share industry- and career-related knowledge and advance the role of women in software testing and technology, consider connecting with the following professional organizations:

The National Center for Women & Information Technology (www.ncwit.org) is a national coalition of nearly 200 prominent corporations, academic institutions, government agencies and nonprofits working to strengthen the IT workforce and encourage technology innovation by increasing the participation of women. NCWIT's work connects programs along the entire pipeline, from K-12 and higher education through industry, academic and entrepreneurial careers.

"The population that creates our technology should be as broad and diverse as the population it serves. Diversity in technology occupations brings different problem-solving approaches, fresh ideas and greater opportunities for innovation. As technology becomes increasingly pervasive in our lives, it's critical that we increase the participation of women in the technology workforce."

– Lucy Sanders, Cofounder and CEO, NCWIT; Bell Labs Fellow

LUCY SANDERS

The Women Technology Council (http://www.womentechcouncil.org/) is a not-for-profit organization providing leadership, support, mentorship, networking and a sense of community for women in all levels of the technology industry, including those from technology companies and those with technology roles in other organizations.

"Women are very important to the software industry. They bring a unique blend of organizational and communication skills paired with tenacity and analytical intelligence. So many areas of software design, development and testing have been improved with the help of many smart women. Our organization strives to be a resource that offers mentoring and guidance for women in technology. We remain dedicated to inspiring interest and recognizing accomplishments for women in technology careers."

– Kimberley A. Jones, Chair of the Board, WTC

Women in Technology International (www.witi.com) was founded by Carolyn Leighton to help women advance by providing access to, and support from, other professional women working in all sectors of technology. WITI started in 1989 as the International Network of Women in Technology, and in 2001 evolved into the WITI Professional Association, the nation's leading trade association for tech-savvy women; today, WITI is the premiere global organization empowering women in business and technology to achieve transformation through technology, leadership and economic prosperity. With a global network of smart, talented women and a market reach exceeding 2 million, WITI has powerful programs and partnerships that provide connections, resources, opportunities and a supportive environment of women committed to helping each other. Along with its professional association of networks throughout the U.S. and worldwide, including Hong Kong, Great Britain, Australia and Mexico, WITI delivers value for individuals who work for companies, the government or academia, as well as small business owners. WITI products and services include: Networking, WITI Marketplace, Career Services/Search, National Conferences and Regional Events, Publications and Resources, Small Business Programs, Research, Bulletin Boards and more.

CAROLYN LEIGHTON

The Association for Women in Computing (http://awc-hq.org/) was founded in Washington, D.C., in 1978 and is one of the first professional organizations for women in computing. AWC is dedicated to promoting the advancement of women in the computing professions. Members include many types of computer professionals, such as programmers, system analysts, operators, technical writers, Internet specialists, trainers and consultants. The purpose of AWC is to provide opportunities for professional growth through networking and programs on technical and career-oriented topics. AWC encourages high standards of competence and promotes a professional attitude among its members. A board of directors represents all the local chapters. AWC supports and encourages networking, both in person and via the Internet. It encourages the formation of student chapters at colleges and universities. AWC is a constituent society member of the Institute for Certification of Computer Professionals (ICCP).

The Association for Women in Technology (http://www.awtsocal.org) is a nonprofit organization dedicated to empowering women in all fields of technology. AWT facilitates networking and career development through informal and professional events. AWT also supports a charitable foundation to provide scholarships for women in Southern California.

Software Test
& Performance
PROFESSIONAL DEVELOPMENT

introduces

STP Public Training

Marriott San Mateo • San Mateo, California

April 19 – 23, 2010

Learn • Refine • Advance

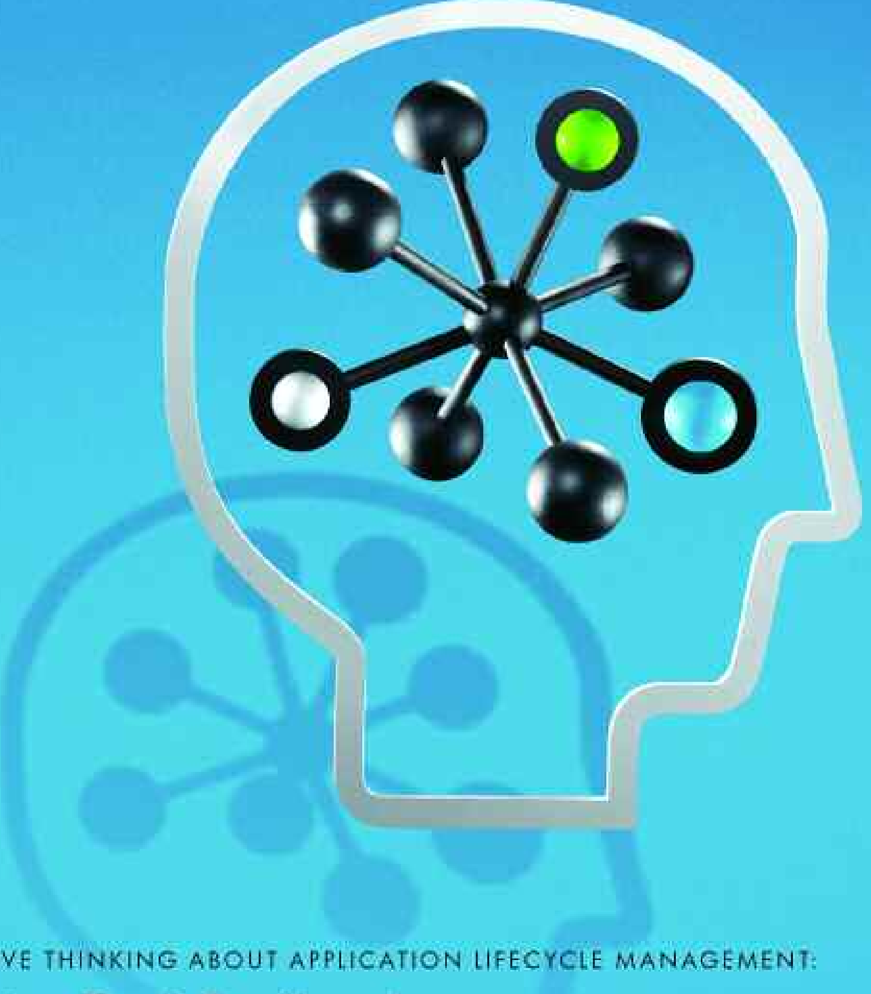
Get in-depth professional education from industry experts! STP Public Classroom Training provides a wealth of benefits, including:

- Experienced instructors
- A structured environment focused on skills building
- Hands-on exercises and real-world examples
- The opportunity to gain peer-to-peer knowledge by networking with other software testing professionals

Save the date & stay tuned for more details!

Join the STP Collaborative Community
to receive special members-only discounts
on all STP Professional Development offerings!

Learn more at www.STPCollaborative.com/Events



ALTERNATIVE THINKING ABOUT APPLICATION LIFECYCLE MANAGEMENT:

Computers Don't Run Your Apps. People Do.

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

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

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

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