July/August 2013 www.TechWell.com

BETTER SOFTWARE™

A TECHWELL PUBLICATION

BUILD A BETTER TEAM
Understand individuals and improve the group

DOWN TO DETAILS
Estimating and Planning Agile Tests

NOT JUST A NUMBER

THE REAL VALUE OF METRICS

POWER-UP YOUR CAREER WITH TEST

The more training you take the greater the savings!

Maximize the impact of your training by combining courses in the same location. Combine a full week of training for the largest discount!

2013 FALL SCHEDULE

TESTING TRAINING WEEKS

September 16–20, 2013
Washington, DC

October 21–25, 2013
Tampa, FL

November 4–8, 2013
San Francisco, CA

WASHINGTON, DC TRAINING WEEK
September 16–20, 2013

Indicates courses pre-approved for PMI PDUs

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
Software Tester Certification—Foundation Level			Mastering Test Design	
Testing Under Pressure	Test Estimation and Measurement	Security Testing for Testing Professionals		Exploring Usability Testing
Systematic Software Testing			Performance, Load, and Stress Testing Workshop	
Essential Test Management and Planning		Leadership for Test Managers	Test Process Improvement	
Fundamentals of Agile Certification		Agile Tester Certification—ICAgile		
			Mobile Application Testing	

WAYS *to* SAVE

Take advantage of the different "Ways to Save" on training using our discount programs listed below. Purchase valuable software quality training for your whole team and save.

EARLY BIRD

Register 6 weeks prior for any training week course and receive $50 off per registered course day. Take a full week of training and save $250!

COMBINE TRAINING WEEK AND SAVE

Combine specialized training courses in the same location and save. Discounts vary depending on the amount of training days combined.

GROUP DISCOUNTS

Have a group and want to save more? Get details on our discount policy by contacting our Client Support Group.

ON-SITE ADVANTAGE

Bring any course to your location for team training. On-site training is both cost-effective and convenient for your team of six or more.

COMBINE CONFERENCE AND SAVE

Save $300 when you combine any our our pre-conference training courses with your conference registration.

For more details on our discount policy, contact the Client Support Group at ***sqeinfo@sqe.com*** *or call 888.268.8770 or 904.278.0524.*

TING TRAINING FROM SQE TRAINING

TAMPA TRAINING WEEK
October 21–25, 2013

Indicates courses pre-approved for PMI PDUs

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
Software Tester Certification—Foundation Level			Mastering Test Design	
Testing Under Pressure	Test Estimation and Measurement	Security Testing for Testing Professionals		Exploring Usability Testing
Risk-Driven Software Testing		Testing with Use Cases	Performance, Load, and Stress Testing Workshop	
Essential Test Management and Planning		Leadership for Test Managers	Mobile Application Testing	
Fundamentals of Agile Certification		Agile Tester Certification—ICAgile		

SAN FRANCISCO TRAINING WEEK
November 4–8, 2013

Indicates courses pre-approved for PMI PDUs

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
Software Tester Certification—Foundation Level			Mastering Test Design	
Testing Under Pressure	Test Estimation and Measurement	Security Testing for Testing Professionals		Exploring Usability Testing
Risk-Driven Software Testing		Testing with Use Cases	Performance, Load, and Stress Testing Workshop	
Essential Test Management and Planning		Leadership for Test Managers	Mobile Application Testing	
Fundamentals of Agile Certification		Agile Tester Certification—ICAgile		
			Test Process Improvement	

LEARNING OPTIONS:

Public — Instructor-led training in a city near you

LIVE VIRTUAL TRAINING — Live, instructor-led classes via your computer

eLearning — Self-paced learning, online

ON-SITE ADVANTAGE — Instructor-led training at your location

For information on our 60+ Public and 40+ Live Virtual Course Dates visit www.sqetraining.com

SQE TRAINING

Test Studio

Easily record automated tests for your modern HTML5 apps

Test the reliability of your rich, interactive JavaScript apps with just a few clicks. Benefit from built-in translators for the new HTML5 controls, cross-browser support, JavaScript event handling, and codeless test automation of multimedia elements.

www.telerik.com/html5-testing

telerik

BETTER SOFTWARE™
A TECHWELL PUBLICATION
Volume 15, Issue 4 • JULY/AUGUST 2013

CONTENTS

in every issue

Mark Your Calendar 4
Editor's Note 5
Contributors 6
From One Expert to Another 10
TechWell Spotlight 12
Product Announcements 33
FAQ 36
Ad Index 37

features

16 COVER STORY
NOT JUST A NUMBER: THE REAL VALUE OF METRICS
Metrics can be enormously helpful, but only if they're used correctly. Abuse them, and they will drive dysfunction. Study the stories behind the data to find the real value.
by Joanne Perold

20 **HOW TO BUILD A SUCCESSFUL TEAM: ANALYSIS, MOTIVATION, AND BEYOND**
It takes more than just gathering a bunch of skilled testers in a room together to make a great test team. Learn how to analyze individual tester types, motivate your team, and achieve success.
by Lloyd Roden

26 **PIECE BY PIECE: TEST ESTIMATION AND PLANNING IN AGILE TEAMS**
The iterative agile methodology provides a clearer vision, smaller time scale, and closer planning horizon. The authors look at approaches to estimation and planning, from product backlog grooming to task-estimating tables and more.
by Ross Collard and Robert Sabourin

30 **USING MISSION AND RISK DIAGNOSTICS TO ENHANCE BUSINESS CONTINUITY**
Developing mission-critical systems can be risky, but an organization can create a better business continuity strategy by combining mission and risk diagnostics.
by Noah Gamer

columns

7 TECHNICALLY SPEAKING
THINGS CHANGE (AND SO SHOULD PROCESSES)
by Jonathan Kohl
Much like the VCRs of yesteryear, our software development processes are not going to last forever. They'll fall out of favor, while new and stronger concepts replace them. Jonathan Kohl writes about coping with process evolution in the quest to improve software.

32 LAST WORD
CAN ANYONE BE A TESTER?
by Julie Hurst
Testers can come from all walks of life, but can anyone be a tester? It's time for testers to clean up the craft and regain their respect.

MARK YOUR CALENDAR

SQE TRAINING

training weeks

www.sqetraining.com/trainingweek

Testing Training Week
September 16–20, 2013
Washington, DC

October 21–25, 2013
Tampa, FL

November 4–8, 2013
San Francisco, CA

Requirements Training Week
September 16–20, 2013
Boston, MA

Agile Software Development Training Week
November 10–12, 2013
Boston, MA

software tester certification

www.sqetraining.com/certification

Foundation Level Certification Training
September 10–12, 2013
Minneapolis, MN
Toronto, ON

September 16–18, 2013
Washington, DC

September 24–26, 2013
Atlanta, GA
Austin, TX

September 29–October 1, 2013
Anaheim, CA

Advanced Level Certification Training
October 28–November 1, 2013
Washington, DC

conferences

STAR*WEST*
starwest.techwell.com
September 29–October 4, 2013
Disneyland Hotel
Anaheim, CA

Agile Development Conference and Better Software Conference East
adc-bsc-east.techwell.com
November 10–15, 2013
Sheraton Boston Hotel
Boston, MA

STAR*CANADA* 2014
starcanada.techwell.com
April 5–9, 2014
Hilton Toronto
Toronto, ON

STAR*EAST* 2014
stareast.techwell.com
May 4–9, 2014
Rosen Centre Hotel
Orlando, FL

Agile Development Conference and Better Software Conference West
adc-bsc-west.techwell.com
June 1–6, 2014
Caesars Palace
Las Vegas, NV

BETTER SOFTWARE™
A TECHWELL PUBLICATION

Publisher
Software Quality Engineering Inc.

President/CEO
Wayne Middleton

Vice President of Communications
Heather Buckman

Director of Publishing
Heather Shanholtzer

Editorial

Better Software Editor
Joseph McAllister

Online Editors
Beth Romanik
Jonathan Vanian
Noel Wurst

Production Coordinator
Donna Handforth

Design

Creative Director
Catherine J. Clinger

Advertising

Sales Consultants
Daryll Paiva
Kim Trott

Production Coordinator
Minda Hatfield

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

As You Wish

In William Goldman's *The Princess Bride*—both the film and the novel on which it's based—the story opens with two characters, Buttercup and her farmhand Westley. She orders him to do chores around the farm, and he obeys every command while saying only, "As you wish." In time, Buttercup comes to understand that by "As you wish," he really means, "I love you." This changes her perspective and gives them both a new lease on life.

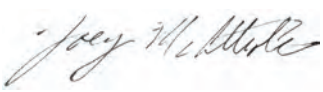
Well, at least until the thieves attempt to kill Westley so the evil prince can marry Buttercup. Otherwise, it would be a very short story indeed.

This issue of *Better Software* magazine features articles in which the authors take a closer look at their subjects. In "Not Just a Number," Joanne Perold describes how metrics are so often abused and explains that it's the stories behind the data that offer real value. Lloyd Roden, in his article "How to Build a Successful Team," teaches us how to analyze different tester types and motivate team members toward success both as individuals and as a cohesive unit. Ross Collard and Robert Sabourin inspect the importance of test estimation and planning on agile teams, and Noah Gamer uses mission and risk diagnostics to improve business continuity.

In the Last Word, Julie Hurst asks a question that could be very important to the craft of testing: "Can *Anyone* Be a Tester?" And Jonathan Kohl, in his Technically Speaking column, suggests that we all spend some time re-evaluating how we do things in "Things Change (and So Should Processes)."

It's all worth a closer look, from methods and processes to teams and even the way we view the software development craft itself. If you've got a story about a time when you looked a little closer and came away with a new perspective, drop us a line and tell us about it.

Yours ruminatively,



Joey McAllister
jmcallister@sqe.com

Contributors

Considered an "expert's expert" by many practitioners, Ross Collard has been called a "Jedi master" by the founding president of the Association for Software Testing. Prominent test professionals say he is an intellectual thought leader with a great wealth of knowledge who has mentored, inspired, and motivated them. Ross was named the Software Test Luminary of 2011 out of approximately one million test professionals worldwide. He has helped his clients and society save billions of dollars through better systems.

Noah Gamer is a driven business leader with experience in Internet marketing, web software development, and security software. Currently, Noah develops Internet strategy and directs global SEO for Trend Micro.

Payson Hall is a consulting project manager for Catalysis Group Inc. in Sacramento, California—and a magician. Payson consults on project management issues and teaches project management. Email Payson at payson@catalysisgroup.com, and follow him on Twitter at @paysonhall.

Julie Hurst is a software tester with almost ten years of experience in the IT industry. Currently, she is working as quality assurance and testing lead for Calgary-based FGL Sports, Canada's largest retailer of sporting goods. Julie is originally from Australia and is passionate about bringing fun into software testing and motivating testers around the world. You can follow Julie on Twitter at @joolery.

Jonathan Kohl is an internationally recognized consultant and technical leader, popular author, and speaker. Based in Calgary, Alberta, Canada, he is the founder and principal software consultant of Kohl Concepts, Inc. Jonathan helps companies define and implement their ideas into products, coaches practitioners as they develop software on teams, and works with leaders to help them define and implement their strategic vision. Read more of Jonathan's work at kohl.ca or contact him at jonathan@kohl.ca.

Claire Lohr has been an active professional in the computer field for thirty years, with the past twenty years emphasizing software process improvement and testing. She currently provides training (design, authoring, and instruction) and consulting services for a wide variety of government and commercial clients. Claire was the chair of the working group for the revision of the IEEE Std 829-2008 Software and System Test Documentation. She holds a BS in computer engineering from Case Western Reserve University.

Joanne Perold is passionate about agile and an ambassador for the people side of software development. She loves learning and finding new ways to solve problems. Being a ScrumMaster at DSTV Online means Joanne gets to put all her passion into practice, watching and helping teams grow. And, in her spare time, she enjoys hiking, riding horses, good food, and great wine.

With more than thirty years in the software industry, Lloyd Roden has worked as a developer, test analyst, and test manager. He was a consultant and partner with Grove Consultants from 1999 to 2011 before establishing Lloyd Roden Consultancy, an independent training and consultancy company specializing in software testing. Lloyd was the recipient of the European Testing Excellence award and has spoken at conferences around the world. His passion is to enthuse, excite, and inspire people in the area of software testing.

Robert Sabourin has thirty years' management and technical experience leading teams of software development and test professionals. A well-respected member of the software engineering community, Rob has trained and coached thousands of professionals. He frequently speaks at conferences and writes on software engineering, testing, management, and internationalization. The author of *I am a Bug!*, the popular software testing children's book, Rob is the president of AmiBug.Com.

Things Change (and So Should Processes)

Software changes, as do software-building processes. When old processes outlive their usefulness, let them go.

by Jonathan Kohl | jonathan@kohl.ca

The other day, I had a fascinating conversation with a team that was struggling with its software development process. This isn't uncommon, but it was sad to see that the team members blamed themselves. They had implemented a popular software development process and, after some initial success, found that they had stagnated and felt like they were regressing. They looked sad, defeated, and hopeless. They used terms like "lack of confidence" and "feeling overwhelmed" and powerful words like "fear" and "hurt."

I felt for them. They had worked hard to implement a software process, and it was letting them down, yet they felt that it was their fault. I asked them, "Do you realize that this process was created twenty years ago? How much has the technology that you produce changed in that time? What about in the rest of the world—not to mention market conditions, people's expectations, and other tools we depend on?" They looked shocked, and then we laughed at the absurdity of it. Of course things were vastly different over that time, so why would the software process be any different?

We are undergoing a massive shift in technology right now—mobile devices, more powerful and ubiquitous wireless networks, and massive computing distribution with cloud technology. Big data is changing how we deal with data from a technical perspective, with enormous ramifications for business. Genetic engineering is helping us discover ways to combat injury and disease, while also creating controversy about the food we eat. Nanotechnology may bring about a giant scientific leap by enabling us to do amazing things in fields like engineering, electronics, energy, and health care. 3-D printing enables us to create and share things in a revolutionary way. The recent proliferation of inexpensive sensors means that pervasive computing can enrich the things we interact with daily.

When is the last time you used a Betamax VCR player from the 1980s? Would you prefer it to a high-definition disc or Internet streaming? It shouldn't surprise us that when we upgrade technology, the older, accepted ways of creating technology will also be disrupted and rendered obsolete. Our favorite process may not apply to what we are doing right now, either as a whole or in part.

Our software development processes are not going to last forever. What worked last year may not work this year, and what worked for the last release may not work for this release. There are a couple of reasons for this. First, we get tired of repeating a routine over and over, and we stop engaging after a while. Imagine eating the same thing for lunch every day, week after week. You are still getting nourishment and the health benefits of eating lunch, but humans are complicated and need variety. Ask anyone who exercises regularly about "hitting a plateau" or a weight-loss regime that is no longer effective. Athletes constantly have to adjust workouts because their bodies grow accustomed to old routines and require something new to continue the process of change.

> **"In regular life, we not only expect a fusion of ideas and a diverse mashup of concepts, we *demand* it because our tastes change."**

The same thing happens with processes we follow. What worked well for us in the past may stop working simply because we get bored and we disengage. Our brains aren't stimulated anymore, and we are just going through the motions.

Another reason an older process may not work is that, sometimes, what has worked very well in the past stops working suddenly. On software teams, this often occurs due to environmental (market or economics, customer expectations, etc.) or technology changes. Our generally accepted practices may no longer be compatible with the new technology we need to use.

Here's a simple example I've witnessed on mobile teams. A generally accepted agile development practice is the extensive use of automated unit tests. During the past fifteen years, we have seen an absolute explosion in unit testing tools, frameworks, practices, and bodies of knowledge in this space. It's been exciting to be a part of it. On mobile development frameworks, though, automated unit testing support can be quite

weak. That is a cause for concern when we are used to the benefits associated with these tools on web or other older technologies. On mobile devices, *state* is incredibly important—wireless conditions, the movement and optimization sensors within the devices, human interaction, emotions and perceptions, and even things like weather changes and lighting. It turns out that automated unit tests don't really address any of those problems.

And yet, the problems that end-users complain about the most involve mobile apps that don't function for them when they are on the move or when they use the devices in different combinations of the states that I mention above. So, my mobile developer friends tend to rely far less on automated unit testing and instead supplement it heavily with other quality practices.

Some of my agile coach friends have almost lost their minds with concern upon seeing that a mobile development team uses few if any unit tests—only to discover that the team uses different tools that *better suit* the quality criteria their end-users require. The mobile team has most likely started out with a standard development process and toolset, found gaps or a lack of support in certain areas, and adapted the process.

While some people might be uncomfortable with this concept, it is natural, and we have examples all around us of how people create something unique by combining ideas and forming fusions or mashups. Many of my favorite musical artists mix influences like traditional blues styles and heavy metal or Eastern music traditions and pop. In fact, there is so much crossover in music now that we would be hard pressed to turn on the radio and hear any one pure style. This also extends to the food we eat, the art we enjoy viewing, the clothes we wear, and our relationships with people from other cultures in an increasingly connected, social world. In regular life, we not only expect a fusion of ideas and a diverse mashup of concepts, we *demand* it because our tastes change.

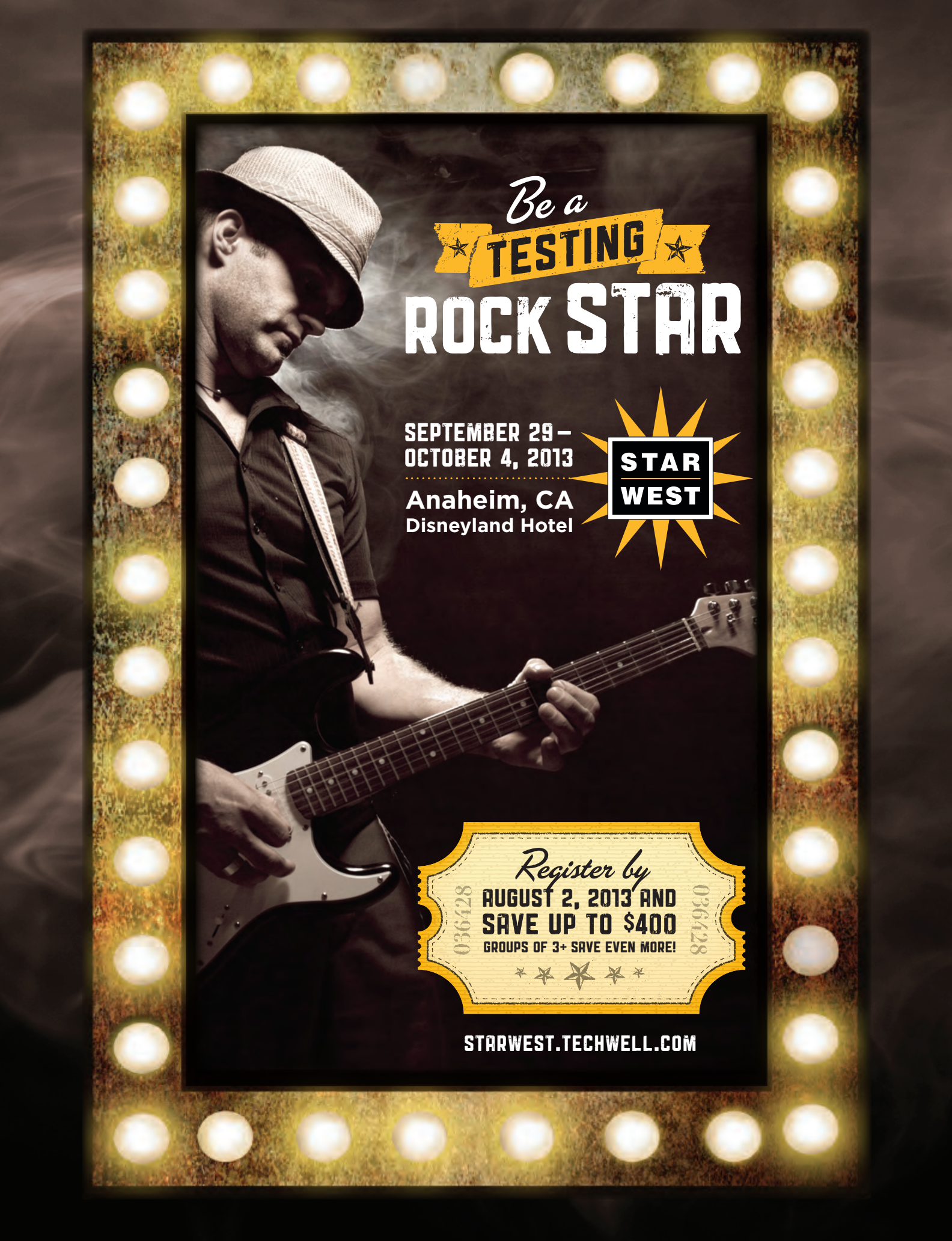
Why, then, do we need to leave our software processes as pristine, unchanging, and implemented exactly the way everyone else seems to be doing it? Shouldn't our innovation also extend to how we create something, as well as what we create?

One of the most painful situations I see repeatedly as a consultant is people who will do whatever they possibly can to bring about some sort of change, regardless of the effectiveness of that change. "If we just got better requirements, or automated tests, or changed to an agile process, or did *X*, then our problems would be solved!" People who are desperate for change fail to realize that the change they so zealously fight for might not be the right solution at this point in time. Or, it might work for a while and then lose its effectiveness. Is it worth it to sacrifice your well-being, your reputation within an organization, and, at worst, your health to get the change through? Be careful about pushing for a change when it just isn't working and people aren't receptive to it. It might be the wrong change.

Another depressing scenario that I see repeated is teams that feel compelled to have process perfection or to follow what is popular at the time, instead of thinking about how their process helps them create technology to promote a great customer experience. Rather than innovate in what they deliver, they feel guilty for doing something different. The software field is vast, with very different mixes of technology, people, culture, and end-users, so why would you expect a process that seems to work for others to work exactly the same for you? I tell teams that they should feel proud to be unique and that they have a different path to follow. They are the true leaders, not the people who want to follow what is popular. If the process is getting in the way of your ability to deliver great software, the process needs to change.

If you have implemented a process and, after initial success, you've found quality issues, people falling behind, or that you are constantly late and over budget, then it might be tempting to think that you are the problem. Rather than blame the people, look at the process. It may not be appropriate anymore. Is it helping all the people on your team to create value, or is it now hindering them? Also, remember that we need to create value not only for our customers, project stakeholders, and company owners but also for our teammates and ourselves. If any of those areas aren't actively creating value, then there is a problem.

Furthermore, don't expect processes that were created when VCRs were popular and cell phones were the size of shoeboxes to fit the technology you and your team are creating now. In this business, we need to change or else we'll fall behind. **{end}**


Be a
TESTING
ROCK STAR
SEPTEMBER 29 –
OCTOBER 4, 2013
Anaheim, CA
Disneyland Hotel
STAR
WEST
Register by
AUGUST 2, 2013 AND
SAVE UP TO $400
GROUPS OF 3+ SAVE EVEN MORE!
036428
036428
STARWEST.TECHWELL.COM

Dale Emery

Years in Industry: **More than 30**
Email: **dale@dhemery.com**

Interviewed by: **Payson Hall**
Email: **payson@catalysisgroup.com**

"So far I've identified fifteen test automation zombies in the wild, and the catalog is growing. They're everywhere."

"Like regular zombies, test automation zombies are relentless and infectious, and they eat your brains. Unlike regular zombies, test automation zombies don't look like people. They look like ideas."

"Each zombie idea involves a tradeoff. If you know you're making the tradeoff, and you make the tradeoff mindfully every time, you can avoid infection and protect your brain."

"In every case, the desire that gives birth to the zombie is perfectly reasonable. We want to keep costs low. We want confidence in our tests. We want to minimize our automated tests' susceptibility to normal variability in system response times."

> "What makes these zombie ideas dangerous is when we apply them by habit or, worse, by institutionalizing them."

"If you see a test automator job title, or a test automation group in an org chart, you know you've institutionalized the idea that test automation is something separate from development. If you create a role to sift through test results, you've institutionalized a barrier between test results and developers."

"The one that scares me most is the Record and Playback Zombie. This zombie is the idea that all you have to do is run through your manual test suite, which you were going to do anyway, and you end up with automated tests."

Opening Acts!

TUTORIALS

PAUL HOLLAND A Rapid Introduction to Rapid Software Testing

HANS BUWALDA The Challenges of BIG Testing: Automation, Virtualization, Outsourcing, and More

DALE PERRY Getting Started with Risk-Based Testing

JENNIFER BONINE Leading Change—Even If You're Not in Charge

JAMES BACH Rapid Software Testing: Strategy

DOROTHY GRAHAM Management Issues in Test Automation

RICK CRAIG Measurement and Metrics for Test Managers

JULIE GARDINER Test Estimation for Managers

PLUS 26 MORE!

STAR WEST

SEPTEMBER 29 – OCTOBER 4, 2013

ANAHEIM, CA
DISNEYLAND HOTEL

STARWEST.TECHWELL.COM ★ GROUPS OF 3 OR MORE SAVE BIG!

Featuring fresh news and insightful stories about topics that are important to you, TechWell.com is the place to go for what is happening in the software industry today. TechWell's passionate industry professionals curate new stories every weekday to keep you up to date on the latest in development, testing, business analysis, project management, agile, DevOps, and more. Here is a sample of some of the great content you'll find. Visit TechWell.com for the full stories and more!

What Makes for a Healthy Software Team?

by Steve Berczuk

Tools and processes are an important part of helping people on a team work together. It's hard to imagine having a team of people collaborate on a software project without some basic tools. An issue-tracking system—even something as basic as an index card on a wall—helps people understand what work is in progress and who needs help.

While a software configuration management system can facilitate collaboration on code and automated tests and continuous integration environments provide rapid feedback on a project's status, there is more to maintaining the health of a team than providing the right technologies and processes.

Continue reading at well.tc/HealthyTeam.

Wading through the Big Data Hype

by Beth Cohen

Is big data really the next big thing, or is it just a way for the big IT shops to scare their enterprise customers into buying more gear and services? While more than 85 percent of respondents to a recent IDG study agreed that big data offered business value, only 23 percent deemed their own projects successful. One begins to wonder if there is less than meets the eye.

Big data is not really new. Companies have been generating and mining vast amounts of information since the dawn of the computer age. While there are plenty of data warehouses in those traditional monolithic systems requiring intense care and feeding, more and more data today is gleaned from multiple distributed sources, formal structured databases, unstructured data feeds, and semi-structured object stores.

What has changed radically is how and where data is stored, consumed, and used.

Continue reading at well.tc/BigDataHype.

The Importance of an Integrated ALM Toolset

by Joe Farah

An integrated application lifecycle management (ALM) toolset—what we're all looking for—is one that has ALM functions working together to provide a well-defined lifecycle process and capability. But beware! Not all integrations are alike.

I'm actually surprised to see the term "integrated" used when tools are almost literally Scotch-taped together. One tool, in its promotional web pages, refers to the "application you've *integrated*"—when all you do is specify the tab label, the location of the application, and the access permissions. While you can then use that tool by clicking on the labeled tab, that's far from integration.

Continue reading at well.tc/Toolset.

Employee Recognition and What Makes It Work

by Naomi Karten

In an IT division meeting at a company where I was once a consultant, one of the managers excitedly announced that two employees had earned the recently implemented "Superstar Award." I cringed—not because two individuals were being recognized for their efforts, but because 187 others weren't.

Management had created this award as a way (so they hoped) of stemming the dropping morale among employees. "Let's show them we're paying attention and we care!" was their thought. So *they* created the award and they decided who deserved it. Bad idea. Singling out two individuals for special attention in this manner runs a risk of making all other employees feel that their efforts don't count. And enthusiastically announcing the award at a meeting at which so many other employees feel taken for granted is a blooper and a half.

Continue reading at well.tc/Recognition.

How Embracing Feedback Helps You Avoid the Vacuum

by Anuj Magazine

Retrospectives are more than a project post-mortem designed to look back at the various happenings along the way of a newly completed job. Their true worth comes from creating an opportunity for members to provide valuable feedback not just to each other but also to the stakeholders on the quality of the project's execution.

Feedback is an often dreaded and misunderstood word. Bill Gates, through a recent TED Talk, focused on this notorious "F" word when he talked about getting teachers the feedback they deserve so they can improve their practice. He opined that a strong foundation of constructive feedback is an absolutely essential tool in athletics, but in education, the practice is almost nonexistent.

Continue reading at well.tc/Vacuum.

Should You Measure Agile Adoption Effectiveness?

by Kent J. McDonald

A question I frequently get when helping organizations transition to agile is "What metrics should we use to measure our agile adoption?" I think this question is premature. People really should be asking, "Should we measure our agile adoption?"

Measurements themselves are not inherently bad and can be useful when used appropriately. So if you can create appropriate measures related to an agile transformation, yes, you

Headliners!

KEYNOTES

by TESTING ROCK STARS


What Executives Value in Testing

Michael Kelly, DeveloperTown, and Jeanette Thebeau, Ex2 Partners

Testing the Xbox: Lessons for All

Alan Page, Microsoft

Lightning Strikes the Keynotes

Lee Copeland, Software Quality Engineering

Selling (and Buying) "Live Site Quality" at eBay

Jon Bach, eBay, Inc.

The Bounty Conundrum: Incentives for Testing

Shaun Bradshaw, Zenergy Technologies

STAR WEST

SEPTEMBER 29 – OCTOBER 4, 2013

ANAHEIM, CA
DISNEYLAND HOTEL

STARWEST.TECHWELL.COM ★ GROUPS OF 3 OR MORE SAVE BIG!

should measure it. The trick is in figuring out what an appropriate measurement is.

Continue reading at well.tc/Effectiveness.

Using Crowd Wisdom as a Marketing Tool

by Rajini Padmanaban

Crowdsourcing in its various forms has become a powerful technique used to connect with the end users and community, to engage with them, and to leverage their wisdom—all to benefit organizations or other community members at large. Crowdsourcing is a blanket term that refers to sourcing or engaging the crowd or community in one of the following ways:

Crowd Creation: Working with the crowd to have them create content for you based on their subject matter expertise in areas such as software development or building photo repositories. Popular examples include Linux and iStock Photo.

Continue reading at well.tc/CrowdWisdom.

The Importance of "Occasional User" Requirements

By Adrian Reed

Organizations in both the public and private sector are increasingly providing consumers with the ability to interact digitally, rather than in person or by phone. For the right type of interactions, online self-service can provide convenience to the customer while also saving the service provider money. The UK government's stance of "digital by default" illustrates the weight that this argument is given.

When processes move to the web, non-functional considerations—including usability—become crucial. There is often a perception that certain sections of the community will simply refuse to use online self-service tools. This doesn't have to be the case.

Continue reading at well.tc/Occasional.

Supercomputer Provides Crucial Information for HIV Research

by Beth Romanik

After years of research into the chemical structure of HIV in hopes of unlocking a way to eradicate the virus, a potential key has been provided by a supercomputer.

Scientists have long been examining HIV's protein casing, or capsid, which protects the virus's genetic material, delivers that material into new host cells, and weakens an infected person's immune system. In short, the capsid is the gatekeeper and propagator of the virus to defeat.

Consequently, the capsid has been the target of antiretroviral drugs, but it is made up of more than 1,300 identical proteins forming an asymmetrical cone-like shape, making it nearly impossible for researchers to unravel the complex subatomic structure.

Continue reading at well.tc/HIVResearch.

Agile Is Not for Everyone (and That's OK)

by Johanna Rothman

Some people claim agile has "crossed the chasm." Certainly, many people are aware of agile. Many people understand that a cross-functional team works in increments, delivering features, asking for feedback. That's at the team level.

The problem is that agile is not just for teams. Once a team installs agile, the team bumps up against systemic management issues. Management has to be willing to change. Program management has to be willing to change. HR has to be willing to change. Finance has to be willing to change. That's huge. We're talking about shifting an organization's culture.

Continue reading at well.tc/Everyone.

Look at Application Lifecycle Management during Project Inception

By Joe Townsend

All projects have a beginning, which could occur in a boardroom, break room, or even in a crowded restaurant. During a project's inception phase, various tasks usually take place. Let's take a look at application lifecycle management (ALM) during this stage of a project and what roles configuration management (CM), quality assurance (QA), and quality control (QC) play.

In the inception phase there are a few items needed from a CM, QA, and QC perspective. 5AM Solutions offers us a list of artifacts that need to be produced during this time, and the company mentions CM, QA, and QC "approaches" as being part of the project plan. The CM approach could be a reference to an organization's CM plan or a more specific project CM plan. The QA and QC approach is the test plan for the project, which is defined further on 5AM's website.

I certainly do not disagree with this approach and agree completely that a CM, QA, and QC plan is needed, and these items should be discussed possibly even before inception. So, do the roles of CM, QA, and QC run deeper in the inception phase or are we simply to provide two documents (CM plan and the QA or QC plan) and wait to be contacted?

Continue reading at well.tc/ProjectInception.

Can Your Agile Team Be Trusted?

by Noel Wurst

I write a lot about successful collaboration among agile teams. The focus is usually the same: The stories often deal with teaching teams how to communicate so that collaboration is made easier and a creativity-inspiring culture can thrive.

The strongest teams that have the most success with collaboration are those where a great deal of trust has been established among members. Mike DePaoli writes that without that trust, "you end up with a group of individual contributors and not a team."

Many agile development teams put a great deal of effort into establishing this trust in house; but what about the trust that must also be built between the development team and the clients?

Continue reading at well.tc/Trusted.

Backstage Pass

NETWORKING EVENTS

Welcome Reception and Karaoke Jam Session
Tuesday, October 1 • 4:30–6:30pm

Expo Reception
Wednesday, October 2 • 5:30–6:30pm

Meet the Speakers at Lunch
Wednesday, October 2–Thursday, October 3 • During Lunch

Bookstore and Speaker Book Signings
Tuesday, October 1–Thursday, October 3

STAR*WEST* Test Lab
Wednesday, October 2–Thursday, October 3

Presenter One-on-One
Wednesday, October 2–Thursday, October 3

STAR WEST

SEPTEMBER 29 – OCTOBER 4, 2013

ANAHEIM, CA
DISNEYLAND HOTEL

STARWEST.TECHWELL.COM ★ GROUPS OF 3 OR MORE SAVE BIG!


NOT JUST A NUMBER
THE REAL VALUE OF METRICS
by Joanne Perold
THINKSTOCKPHOTOS.COM

Metrics have their place and can be extremely useful. They can also drive dysfunction when used in the wrong way. The minute you assign a number to something, people begin to focus on the number itself and not on the story that it is telling or the information being provided. But the stories and the context behind the data often are far more valuable.

Velocity as a Measurement

I had a team with a fluctuating velocity. Over a period of eight sprints, they were not even vaguely consistent, and the product owner therefore found planning difficult.

I decided to have a look at the detail and keep track of the information behind the numbers. The team was consistently losing and gaining team members due to contractors moving backward and forward for visa renewals and requirements and team members' going on leave. The team size was never consistent, so how could their velocity possibly be? This number was not entirely useful on its own for the team, but it helped determine that communication and interacting when team members were abroad were huge issues. It was very useful as a starting point for trying to solve the problem of difficult communication.

How Does Velocity Become Dysfunctional?

I have heard in agile and not-so-agile environments things like "The team needs to double their velocity in *n* sprints." What I do not understand is how anyone can fail to see the dysfunctional behavior waiting to happen when a team is tasked with this.

The behavior I have observed frequently in this instance is the team's doubling its story points per story. Every three becomes a five, and every five becomes an eight. The only thing this achieves is turning an insightful metric into a number that adds little or no value. Most unfortunately, this can affect the team's trust and behavior—both the trust that team members feel from management and the trust that they have in management.

The lowest tier of Patrick Lencioni's *Five Dysfunctions of a Team* is the absence of trust. [1] In my experience, using the metric in this way can start a vicious, dysfunctional circle with a lack of trust at its core.

Bugs as Information

Information on bugs can be very useful for both testers and other teams. The status of a product's health should include how many bugs have been found and which ones the testers and product owners feel are non-negotiable for launch.

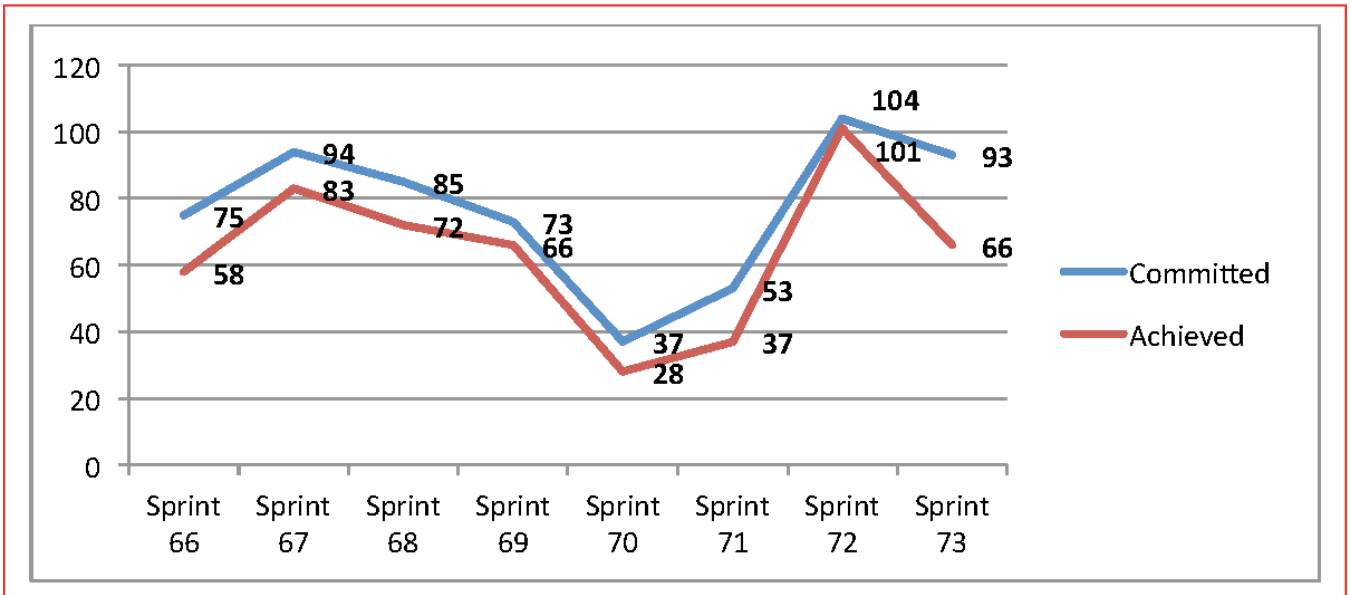


Figure 1: The pattern across sprints—especially sprints 70 and 71—shows a lack of team direction.

Another problem highlighted by looking deeper into the meaning of the metric was that the team had a distinct lack of direction. The backlog was inefficiently groomed and prioritized and occasionally almost nonexistent. You can see the effect this had in sprints 70 and 71, which were particularly bad (see figure 1). Once the team members were aware of this, they put practices in place to ensure more visibility and encourage better stories and prioritization from the product owner.

Being exposed to this information was invaluable for the team. It helped them to ask questions about what was happening and what they needed to improve going forward. The individual numbers themselves were fairly worthless, but the patterns were hugely insightful.

When reviewing the number of new bugs in sprints, the sprints with more bugs might mean that the team was pushing too hard for speed and quality slipped. By looking at the kinds of bugs and the numbers of bugs, the team can determine the value of the ratio of quality to speed. If this ratio is something that the team, testers, and product owner are happy with, then all is good. If not, then a new negotiation needs to take place.

The average time it takes a critical bug fix to go live can also be valuable information for a team. If a release is difficult, takes a long time, and the team has to release the past

fifteen features and regression test for two weeks before they can release, then maybe there is a problem. If teams have this information and understand the impact, then they can work on solving the problem.

Knowing the overall health of the build or product is far more useful than knowing that we have ten unresolved bugs. Those ten bugs could be symptoms of something serious within the system, or they could be minor text and UI changes that need to be implemented.

"Knowing the overall health of the build or product is far more useful than knowing that we have ten unresolved bugs."

What's Wrong with Bug Counts?

I was speaking to the product owner of an interesting project. He was complaining about the number of bugs logged that were essentially duplicates. I asked how the testers were measured, and he said bug counts.

What scared me most was that the metric had encouraged behavior that put the focus completely on the wrong things. The testers were focused on easy-to-find, UI-visible bugs that could be logged and counted separately per screen or per event. They were not at all focused on getting information about product health as a whole. The testers were not looking for information about what we do not know—the tests that take time and patience and intricate thinking to invent and run. Those tests provide valuable information if they yield any issues.

The metric, especially when used as a measure in this way, was totally driving the wrong behavior.

What Now?

Metrics come in all shapes: velocity; bug counts; code coverage; cycle time; live releases per sprint, week, or month; etc. These can all provide valuable information for creating visibility around things that team members do not know. If the focus of the metric is only on how much or how many or how often, then I believe the value of the metric for the team decreases. The wrong behaviors are rewarded and encouraged, and too often trust levels are affected. In my experience, it is far more useful—and far less dysfunctional—to pay attention to the meaning and the nuances of the metrics than the numbers themselves. {end}

Joanne.Perold@dstvo.com

Sticky Notes

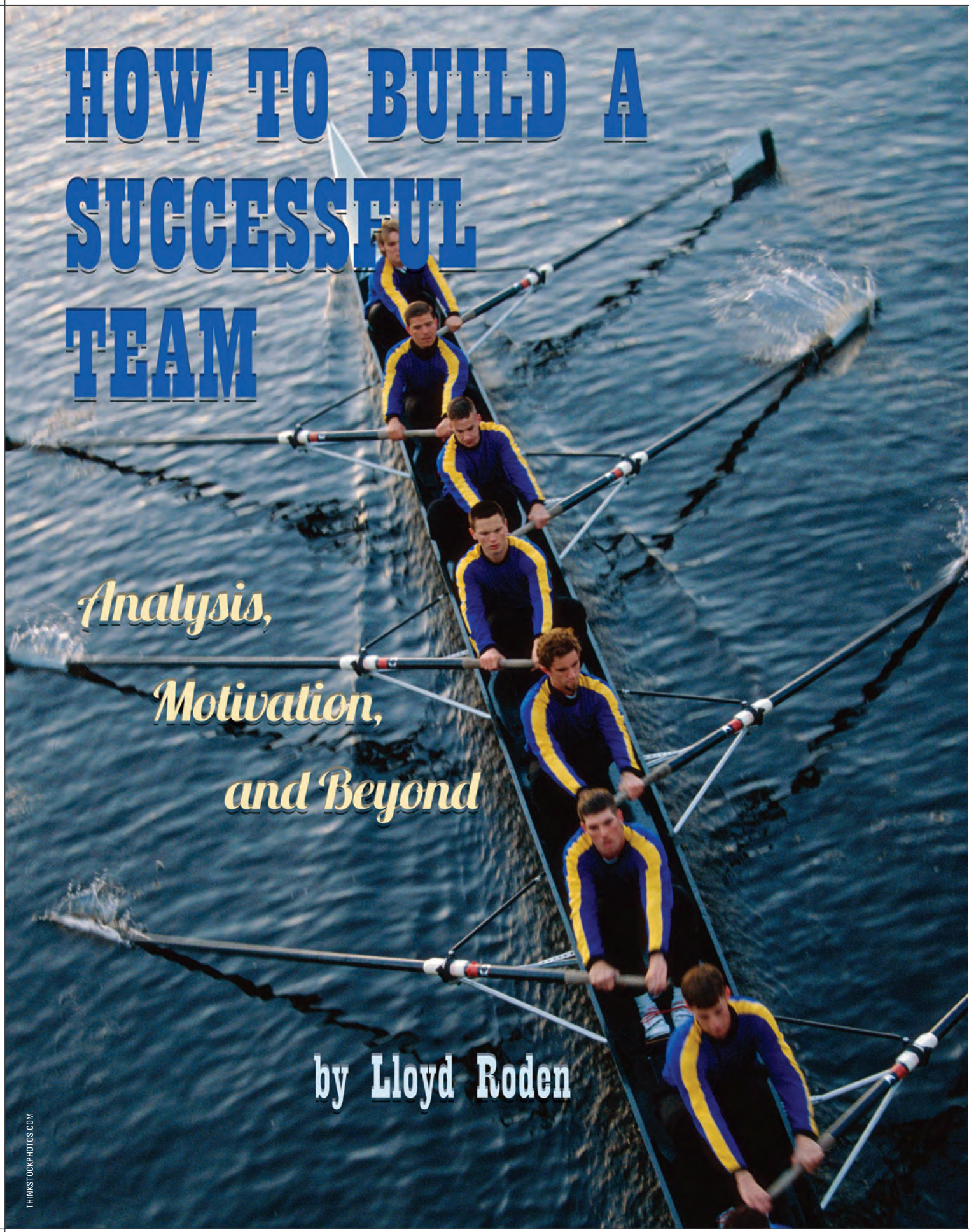
For more on the following, go to StickyMinds.com/bettersoftware.

- References

HOW TO BUILD A SUCCESSFUL TEAM

Analysis, Motivation, and Beyond

by Lloyd Roden



THINKSTOCKPHOTOS.COM

Teams exist everywhere: football and soccer teams, Olympic teams, baseball teams, orchestras, work teams, and more. All of these teams have one thing in common: They consist of individuals with different skills and abilities who come together to achieve a common goal. Sports teams want to win, orchestras want to play incredible music, and work teams want to succeed in whatever they set out to do.

In order to build a successful team, we must first understand what makes a team and how it becomes great.

Know Your Team

In their book *Peopleware*, Tom DeMarco and Tim Lister describe how we can create an atmosphere for what they call "jelled teams"—teams that produce success and productive harmony. The following are some of the common distractions involved in building these jelled teams.

Physical Separation

This is becoming a major problem because modern teams often are not collocated but distributed, often across many different countries. The greater the separation, the more problems can transpire between the teams. When possible, it is important to locate the testers, developers, and designers as close together as you can.

Being Unfair

It is human nature to crave fairness. Children seek fairness from their parents. I have two children and even now, when both of them are adults, I still need to be fair. Unfairness generally means showing favoritism to one group or person above the others. This can manifest itself in a number of ways: salary, remuneration, office space, paid overtime, and other financial and non-financial rewards.

Communication Breakdown

Good communication is one of the key ingredients in any relationship. It is essential to communicate effectively and regularly with all members of the team, keeping them informed about progress, decisions that have been made, and information that will be important to them.

No Clear Direction

The test team must have a vision—clear direction from the management—in order to be productive. Otherwise, team members will end up heading in different directions. State the objectives of the test team and seek senior management's approval of these objectives. The following are some examples of good test objectives:

- Assess software quality.
- Help achieve software quality.
- Assess and report on risk.
- Help preserve software quality.
- Provide timely, accurate, and predictive information.

Failure to Appreciate

Everyone wants to be acknowledged and appreciated for the work they do within the organization. If we do not appreciate one another, then people feel as though they are worthless. Take time to watch your team and praise team members for good work. It is important that this activity is not forced but is born out of a natural relationship within the team. Rewards are good, but even a simple "thank you" or "well done" goes a long way!

Adopting a Blame Culture

From a very early age, we like to blame. When one child does something wrong, the tendency is to blame the others—usually to avoid punishment!

Why do we blame? Do we want others to feel bad, or is it that we want to try to feel better about ourselves rather than taking responsibility? Why is a blame culture unhealthy for productive test teams? Because we become fearful of taking any risk in case we make a mistake. If we are to learn, progress, and become more productive as a team, then we must fight the "blame culture" mentality.

Remember: If you point your finger at someone else, three fingers are pointing back at you.

Failure to Recognize Individuality

Teams are made up of individual people with individual skills and individual personalities. Good managers will recognize the skills that each team member brings and will fully utilize these skills so that the whole is greater than the sum of the parts.

Analyze Your Testers' Styles

To help us understand individual skills and strengths, I have created the Tester's Style Analysis Questionnaire (see figure 1). This will help you determine what type of tester you are and what other tester types you are best suited to work with. I have noticed that there are four types of testers that exist within our organizations. The questionnaire helps us assess these types and the type of testing work they enjoy and are best suited to undertake. This enables us to manage teams more effectively for greater productivity and staff retention.

How to Complete the Questionnaire

Completing the questionnaire is very straightforward. There are no right or wrong answers, but it is essential that you relate your answers to your work life rather than your personal life.

There are two steps:

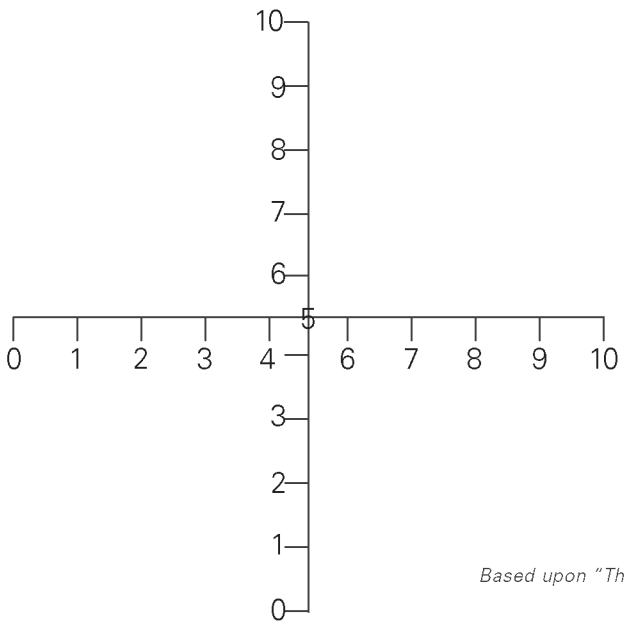
1. For each axis, X and Y, mark a selection from one of the columns. For example, I believe I am more "friendly" than "formal." Therefore, I would put an "X" against "friendly."
2. Once you have selected from each pairing, count your marks in the left-most column for each axis to get an X and Y coordinate to plot on the grid. For example, I score 9 on the X-axis and 9 on the Y-axis, so I plot (9, 9), which I would indicate in the top right quadrant on the grid.

Tester's Style Analysis Questionnaire

Name _____

X-Axis				Y-Axis			
Friendly	___	Formal	___	To the Point	___	Indirect	___
Approachable	___	Retiring	___	Challenging	___	Accepting	___
Casual	___	Businesslike	___	Quick	___	Leisurely	___
Open	___	Guarded	___	Insistent	___	Thoughtful	___
Unstructured	___	Organized	___	Lively	___	Relaxed	___
Sociable	___	Introvert	___	Impatient	___	Patient	___
Intuitive	___	Logical	___	Adventurous	___	Cautious	___
Random	___	Focused	___	Confronting	___	Receptive	___
Warm	___	Cool	___	Competitive	___	Co-operative	___
Perceptive	___	Insensitive	___	Strong Minded	___	Analytical	___

GRID



Based upon "The Communication Styles Analysis"

Figure 1: The Tester's Style Analysis Questionnaire will help you determine your tester type and what tester types you are best suited to work with.

What Does This Mean?

Each grid coordinate represents a testing style: top left is the pragmatist, top right is the pioneer, bottom left is the analyst, and bottom right is the facilitator.

The *pragmatic tester* likes strategy and goals, positivity, results and brevity, practicality, efficiency, and tasks. However, the pragmatist dislikes indecision, vagueness, time wasting, and being unproductive. The pragmatic tester will:

- Be good at setting and monitoring short- and long-term goals for the team.
- Be good at documenting factual test reports.
- Remain positive through pressure.
- Be keen to adopt a "Most Important Tests First" principle.
- Be a strong driving force to ensure a task is done.
- Want to implement efficiency into the team.
- Be self-motivated and task-oriented.
- Make quick decisions.
- Enjoy challenging testing tasks.
- Be content to work alone.
- Be brief and often want some form of structure.

"As managers, we must recognize the strengths of the individual team members as well as the strengths of the team as a whole."

The *pioneering tester* likes new ideas, change, openness, results and efficiency, involving others, and risks. However, the pioneer dislikes standards, detail, "the norm," and paperwork. The pioneering tester will:

- Be good at ad-hoc testing, bug hunting, error guessing, and exploratory testing.
- Be good at challenging and improving things to make them more efficient and effective.
- Have good ideas.
- Be good at brainstorming test conditions.
- Prefer agile methodologies.
- Share ideas about different ways to approach testing.
- Identify and take necessary risks when required.
- Have creative test ideas for how to find more faults.
- Be keen on making improvements and changes.
- Want to involve others.
- Be flexible and adaptable in changing circumstances.

The *analytical tester* likes accuracy, attention to detail, proof, standards, reliability, and all alternatives. However, the analyst dislikes new things and change, untested things and risks, brevity and speed, and letting go. The analytical tester will:

- Be good at defining and documenting test cases.
- Be good at producing test standards and procedures.
- Analyze problems and find the root cause.
- Produce work that is accurate and complete.
- Enjoy writing detailed test documentation.
- Provide proof when faults are found.
- Document thorough test reports.
- Prefer more sequential methodologies.
- Complete work regardless of what it takes.
- Challenge requirements.
- Want to consider all alternatives before making a decision.

The *facilitating tester* likes networking, positivity, being team-oriented, consensus and sharing, building bridges, and status quo. However, the facilitator dislikes pressure and deadlines, confrontation, isolation, and being dictated to. The facilitating tester will:

- Prefer an iterative lifecycle.
- Often ask for opinions before raising issues.
- Be good at documentation.
- Cooperate well with other departments.
- Often see the "other side."
- Be good at defusing "Us vs. Them" syndrome.
- Be popular.
- Make things happen.
- Provide support in testing to other team members.
- Enjoy networking and talking to people.
- Be a team player and want to involve others.

Our Mode of Operation

Although we have put a mark on the grid, we usually operate within a small boundary around the mark and can fluctuate a small amount depending on the circumstances. The more prominent we are within the grid—i.e., the closer to the extreme corners—the more difficult it is to adapt to one of the other styles and take on that particular characteristic. For instance, my score is (9, 9), so I am a very prominent pioneer and therefore find it particularly difficult to document things.

If we find ourselves on the line or in the center of the grid, then we are very adaptable within the styles that we border. However, we can be very difficult to manage. Imagine that you are in the center of the grid and can easily adapt to any style. One day, you could be given a task to produce some procedures. As an analyst you welcome this task, but as a pioneer you dislike it.

Diagonal Opposites Repel

The key in using this analysis questionnaire is to understand where we are and also where the rest of the team members are so that we can assign work that will complement everyone's style. This does not mean that pioneers do not need to produce documentation, but they will find it more challenging than the analytical tester.

It is also worth pointing out that diagonal opposites repel, and this could be a cause of team tension. For example, "John" is a pioneering tester and wants new ways of doing things to help with efficiency. He is constantly challenging standards and procedures. However, "Chris" is an analytical tester and requires structure in order to work efficiently. Chris often challenges John for his cavalier approach to testing, and tension often exists between them. Is this wrong? No, it is just that they are different.

As a general principle, analysts and pragmatists tend toward task issues, while facilitators and pioneers tend toward people issues. As managers, we must recognize the strengths of the individual team members as well as the strengths of the team as a whole. We must also seek to motivate the team and the individuals.

"No one ever said on his deathbed, 'I wish I had spent more hours at the office.'"

Motivating Your Team

There have been many studies and theories regarding motivation, such as Maslow and Hertzberg's theories. In essence, motivated testers will be more productive, efficient, and happier at work. It is therefore in our interest as managers to recognize key signs of motivated and demotivated staff.

Signs of motivation:

- High performance
- Drive and enthusiasm
- Cooperation in overcoming problems
- Keenness to achieve results
- Accepting responsibility
- Working long hours
- Happiness and enjoyment
- Welcoming change

Signs of demotivation:

- Apathy and indifference
- Dissatisfaction
- Poor time keeping and high absenteeism
- Resisting change
- Exaggeration of disputes
- Lack of cooperation
- Blaming
- Withdrawal

Key Motivators for Testers

Having been in the IT industry since 1983, I have come to recognize six fundamental motivators for testers:

Support your team. It is important that test managers listen to the team and, if the need arises, fight in the tester's corner. Discuss various courses and staff development with your team. Think about sending them to testing courses, conferences, or seminars to improve their skills. It is also important to strike the right balance between hands-off and hands-on test management. Small things like sitting *with* the testers rather than isolated in a large office will motivate your team. Test managers should also test the product, as it shows them that you are involved rather than removed.

Provide a variety of testing work. The human brain needs to be stimulated for optimal functionality and productivity. A common saying is, "Variety is the spice of life." If we continually do things in the same way, then boredom will take effect. We must provide a variety of testing work to challenge our team and also provide opportunity for creativity. Examples could be bug-hunting afternoons, paired exploratory testing sessions, brainstorming sessions, etc.

Use tools to help and not hinder. Rather than "test automation," I much prefer to think about "tool support for testing." Tools are there to enable testers to do the work they are best suited for—finding bugs. They should make the tool users' lives easier and more enjoyable. If a tool provides additional work, then it could be the wrong tool.

Provide a mentoring program. We must learn lessons from more mature industries, such as construction, plumbing, electrical, etc. These industries provide valuable apprenticeship programs where junior staff work alongside senior staff to learn the trade. This also happens for lawyers, pilots, doctors, and nurses. However, we often have the "sink or swim" approach for testing, where new staff members are provided with some basic training and just told to test.

Provide some slack. Evidence suggests that employees who are provided slack time become more productive. However, we often ask our teams to work harder and longer hours. This attitude is counterproductive and can have a very negative effect. No one ever said on his deathbed, "I wish I had spent more hours at the office."

Help create a unique selling point for individuals. We must create a valuable and unique selling point for each of our team members and encourage them in this point. Why should we use a particular person in preference to someone else who might be "cheaper"? What makes that person different? By encouraging team members in their unique selling points, they become highly motivated individuals—people who enjoy working for you and will often be loyal to you and the organization.

Conclusion

In order to retain good quality testers within our test teams, we must spend time developing the people side of test management. Recognize your team members' current strengths, and try to address their skill gaps. Learn what strengths and skills each person provides for the team and how you can motivate team members so they can achieve their best as individuals. **{end}**

lloyd@lloydroden.com

High-speed, large-scale **automated** data validation!

Data Test Professional - Get Smart Data Testing

Helps you accomplish end-to-end data testing in 4 simple steps:

1. **Configure** — Source and Target Data Storage
2. **Define** — Test Scenarios for Data Transformation
3. **Schedule** — and Execute Test Scenarios
4. **Analyze** — Summary & Detailed Test Rules

Data Test Professional provides an intuitive interface to configure, schedule and execute tests that makes it easy for any user.

Features

- Automated ETL transformation testing
- Supports multiple data forms & file formats (XML,XLS, TXT) & all JDBC formats
- Platform Independent (Windows, UNIX & MAC)
- Reusable library of SQL queries and data format templates
- Mechanized Batch Scheduler
- Customizable Test Reports (PDF,HTML,EXCEL)
- Seamless Integration with Test Management Tools (HP,IBM & Open Source)

'Achieve up-to **80% cost reduction** with faster time to release'

You may also be interested in,

Automating Automation

Intelligent automated test script scheduling and quality engineering solution

Get Authorized

Validated electronic signature for authorizing testing artifacts stored in HP-QC/ALM, ensuring regulatory compliance

Ultimate Test Design

ADPART is an innovative model based test design solution for real-time business requirements.

Make Scripts Perform Better

One-click generation of performance scripts based on your standards while facilitating easy reusability and review

For more details on our products and their features, please reach us at **QASuite@cognizant.com**

Cognizant™

PIECE BY PIECE

Test estimation and planning in agile teams

by Ross Collard and Robert Sabourin

THINKSTOCKPHOTOS.COM

Many of our customers blend Scrum and kanban frameworks when organizing their agile development projects. Every implementation is different, making it hard to generalize about where agile test estimation fits in. Test planning typically is not treated as a distinct, separate activity, but rather testing tasks are planned as part of the work done to fulfill requirements. Estimation takes place during backlog-grooming meetings and sprint-planning sessions.

Traditional software development lifecycle projects often are sufficiently large and complex that estimation becomes a point of contention. Estimating the productivity and failures of dozens of people you've never met can be hazardous. Waterfall estimation can be top-down, and it can be back-to-front—for example, when we are scheduling backwards to fit the work into a known deadline or a budget constraint. Or, estimation can be bottom-up, rolled up from lower-level components of estimates. These estimation techniques tend to have to assume observations at five meters from ones made at 50,000 meters.

In our experience, iterative agile projects produce better estimates than traditional waterfall development. During both product backlog-grooming meetings and sprint-planning sessions, the vision is usually clearer, the time scale smaller, and the planning horizon closer. Continuous improvements are expected and accommodated, rather than resisted. Quick feedback loops and refinements of project goals and deliverables are built into agile processes. Communication tends to more immediate, open, and informal. Estimates are developed more quickly and accurately, with less effort, and politically are easier to revise.

Product Backlog Grooming

Product backlog grooming involves the careful review and improvement of product requirements well in advance of their implementation. The product backlog is a prioritized list of requirements including user stories, features, non-functional aspects of the system, product deliverables, and constraints. Lower-priority backlog entries have less detail, while higher-priority backlog entries have more detail.

Grooming sessions help the team elicit some detail from the product owner, customer, user, or business analyst. When a backlog entry is groomed, the team develops a common vision of what the requirement is and why it is important to the customer. During a session, the team elicits acceptance criteria for each story. The acceptance criteria often are stated as a series of examples from different perspectives showing what a successful implementation would look like.

During grooming sessions, the team estimates the size of high-priority stories with a unit of measure called a *story point*. A size is assigned to each backlog entry proportional to implementation effort. Story points are team specific and relate to real, recent, relevant experience.

To come up with an estimate, the team plays a few hands of *planning poker*. First, each team member receives a set of cards with possible sizes: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, and ?. Second, the product owner reads through the story being sized. Third, team members vote for the size they feel fits best based on their past experience. Fourth, differences are discussed to resolve disagreement and build a better understanding and consensus. The team members who suggested the smallest and largest sizes advocate their estimates. It can take a few rounds of poker to size a story. Finally, the estimate is completed when all team members are within one card value of each other in the size estimate. The higher value is selected as the size estimate for the backlog entry.

After a few sprints, the team's effective burn rate measured in story points per day should be known within a tolerable range of uncertainty. This burn rate is often called a team's *velocity* and is used as a basis for downstream product estimation and planning. For example, if a team can implement fifty story points per sprint, then the product owner can estimate that within ten sprints the team can probably implement about 500 story points' worth of functionality. Velocity measures a team's productivity.

Estimating Guidelines—Horizon vs. Level of Detail

We usually have less information about tasks that are further into the future. The relationship between the time horizon (how far the planning looks ahead) and the level of task detail (sampled from recent projects) can be seen in table 1.

At the beginning of each sprint, the agile team holds a planning session to identify work required to implement high-priority backlog entries. During the planning meeting, the team identifies technical work required to implement each backlog entry. The team

Estimate Characteristics	Planning Horizon	Task Detail
Waterfall projects		
Midsize/Large	8 to 24 months	12 to 4 person-months
Small	4 to 8 months	2 to 1 person-months
Agile projects		
Low-priority backlog	2 to 6 weeks	8 to 4 person-weeks
High-priority backlog	1 to 4 weeks	4 to 1 person-days
Current sprints	2 to 10 days	8 to 4 person-hours

Table 1: The relationship between the time horizon and the level of task detail. *We reversed the sequence of entries in the task detail column—e.g., "8 to 4 person-hours", instead of "4 to 8 person-hours"—in order to highlight the inverse relationship between planning horizon and task detail.

considers activities related to designing, programming, testing, documenting, and training, and the team and product owner define what it means to be "done." For example, a story might be done when:

- The story is coded.
- All story code is integrated into the product's main code line.
- The story programming tasks are completed.
- The story data-related tasks are completed.
- The story testing tasks are completed.
- The story documentation tasks are completed.
- The story training-related tasks are completed.
- Story-related unit tests pass.
- Story-related acceptance tests pass.
- Product regression unit tests pass.
- Product regression story tests pass.
- Story non-functional acceptance criteria are confirmed.
- Product performance is acceptable.
- The product is robust as demonstrated by passing stress tests.
- The product can be installed.
- Customer data can be migrated from the previous version.
- Auditable evidence exists of regulatory compliance testing.
- All known bugs in the product are acceptable to product owner.

The scope of work can radically change the story size. Without a uniform definition of "doneness," underestimates by factors of one to three or worse are not uncommon. (Special application of Murphy's Law: In an orderly, rational world, underestimates and overestimates should balance out. In practice, all incorrect estimates seem to be underestimates. Sabourin's thesis: The problems of underestimating are most visible only when they will prove to be most embarrassing.)

The entire team collaborates to identify the approach that will be used to implement the backlog entry. Very often, a backlog entry is defined by a user story that involves changes to several tiers of an application, including the data layer, business logic, and presentation layer. Different design alternatives are discussed from the perspective of benefits and risks.

When the approach is chosen, the team itemizes tasks required to implement the backlog entry, conforming to all acceptance criteria and meeting the team definition of "done." Team members with testing skills have a lot to contribute to this discussion and are urged to identify specific tasks. The types of testing tasks for each backlog entry include infrastructure related, data related, customer facing, non-functional testing (experiments or study), robustness, exploratory test charter, regression related, development facing, and business rules related.

Testing Task Type	Minor	Moderate	Major	Extreme
Infrastructure related	4	8	10	20
Data related	2	4	8	16
Non-functional test	2	4	8	12
Robustness	2	4	8	16
Exploratory test charter	2	4	8	16
Regression related	2	4	8	16
Development facing	2	4	8	16
Customer facing	2	4	8	16
Business rules related	2	4	8	16

Table 2: Baseline estimates taken from similar projects.

Some agile teams assign an effort estimate to each task, showing the amount of work in person-hours needed to complete the activity. Teams usually use an activity granularity of about one-half day to two days per task. The ScrumMaster can use the estimate to track effort to finish as tasks are completed. During the sprint, the ScrumMaster tracks the remaining effort and takes steps to ensure work allocation is well balanced between the different team members. Because of the relatively rapid rate of iteration, it frequently is unnecessary to re-estimate the initial estimate within an iteration. Instead, the calibration is done from iteration to iteration. A table of estimates that at the beginning of the project are, let's say, 80 percent derived from external experience is gradually replaced with the project's own internal data as confidence grows. The deltas among estimates and actual efforts should converge nicely after a few iterations. Of course, if the deltas among estimates and actual efforts diverge, it is back to head scratching and Murphy's Law.

Example of a Task-Estimating Table

An electronic data system team began collecting estimation data to help improve task-effort estimations during agile projects. As the project continued, they refined the accumulating data, merging calibrated data based on actual efforts as observed during each project sprint. The team started by using the baseline estimates in table 2, taken from previous similar projects. Note that estimates are hours per task and organized by the type of testing task. Complexity is assigned four levels: minor, moderate, major, and extreme. Although these are subjective assessments, the team can reference recent relevant examples of each task's complexity levels.

During each sprint, the EDS team tracked the actual hours required per task and updated the baseline values with their project averages (see table 3). The ratios in the merged results were derived from the team's familiarity, confidence, and relative stability of the data.

Other Agile Estimation Approaches

Here are some alternative estimation approaches we have encountered. Teams often combine one or more of these techniques.

T-shirt sizing: Backlog entries are classified as being small, medium, large, or extra-large.

Size complexity: The team assigns a size and complexity to each story. Low-complexity entries are extensions of existing behavior. Medium complexity involves implementing new code or algorithms. High-complexity entries involve many variables and addressing new concepts. Size and complexity are used to

Testing Task Type	Minor	Moderate	Major	Extreme
Infrastructure related	3	4	8	16
Data related	6	2	4	6
Non-functional test	0	2	4	8
Robustness	0	2	4	8
Exploratory test charter	0	2	4	8
Regression related	4	2	4	4
Development facing	0	2	4	8
Customer facing	4	2	4	8
Business rules related tasks	8	6	8	10

Table 3: Actual effort required per task in hours.

estimate effort based on past data.

Three-point estimates: When estimating a task's effort, instead of just estimating one value, the team estimates three values: the optimistic value, or what effort would be required if everything went perfectly following the so-called happy path; the pessimistic value, or what effort would be required if Murphy's Law kicked in and everything that could go wrong actually went wrong; and the most likely value, or what effort is typically required to complete this task. Once the team has these three values, a formula called the program evaluation and review technique (PERT) formula can be used to come up with the team's task-effort estimate:

$$Effort = \left(\frac{optimistic + pessimistic + 4\times(most\ likely)}{6}\right)$$

The three-point estimation approach is popular among members of the Project Management Institute and is explicitly referenced in their *Project Management Body of Knowledge*.

Some Concluding Remarks

Test estimation in agile teams is tightly coupled with planning. Groomed backlog entries with clearly defined acceptance criteria should be sized by the team responsible for implementing them. Agile planning sessions are collaborative team activities in which design and implementation alternatives are explored and a task list is created describing the actual work to be done. When tasks are identified, teams can estimate their respective effort. When testing activities are broken down into tasks, any suitably skilled and qualified team member can do them. Breaking down tasks to implement stories enables teams to estimate and track the effort required to implement backlog entries and redistribute work across different team members. **{end}**

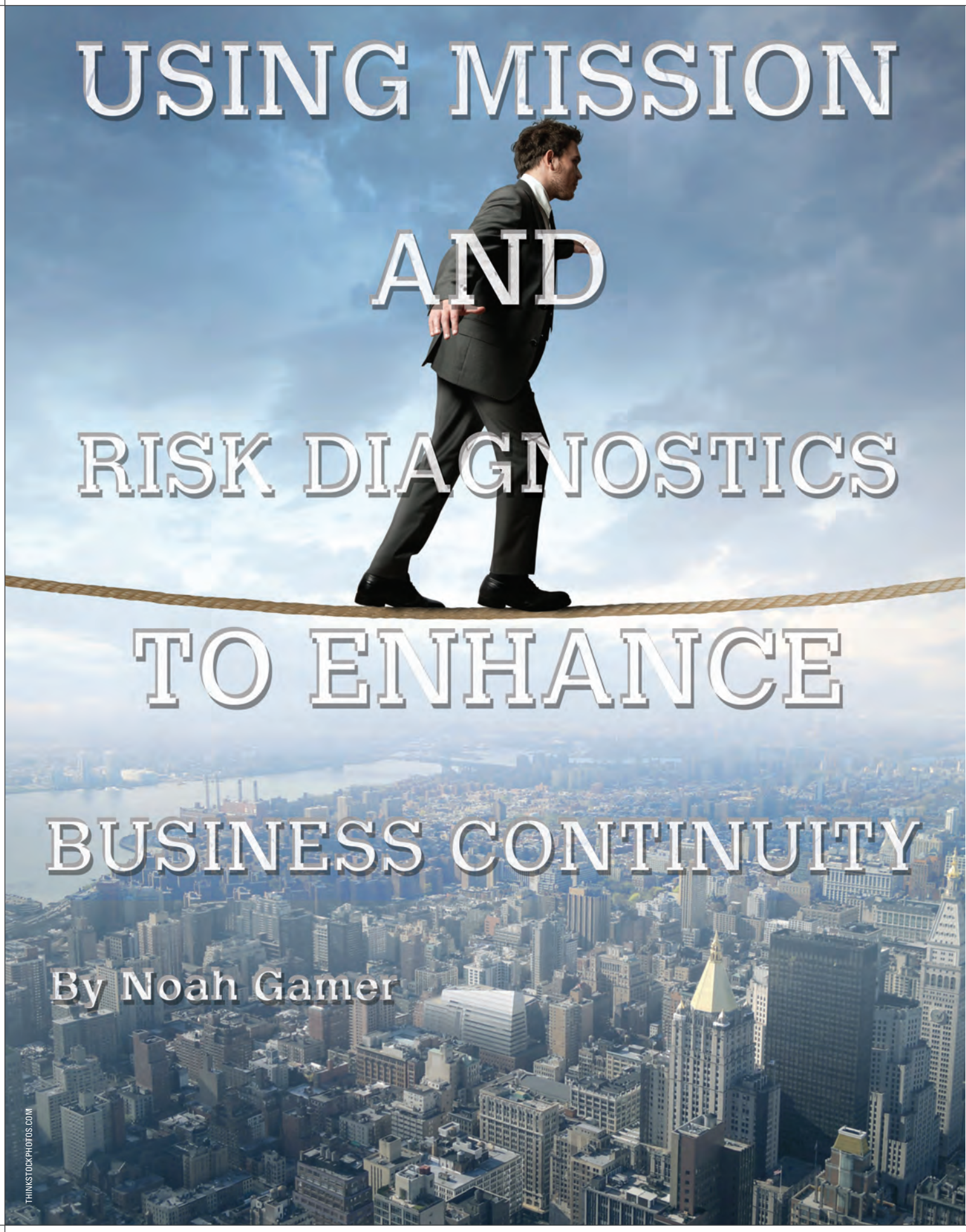
ross@rosscollard.com
rsabourin@amibug.com

Sticky Notes

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

- Further Reading

USING MISSION AND RISK DIAGNOSTICS TO ENHANCE BUSINESS CONTINUITY



By Noah Gamer

THINKSTOCKPHOTOS.COM

"Make sure that each employee or contractor knows who is accountable for each risk metric, and clearly define each person's role."

Developing mission-critical systems can be a risky business, and many large-scale projects have failed due to unforeseen risks like market factors and new technology. In 2009, Lockheed Martin partnered with the Software Engineering Institute (SEI) to develop a risk management plan based on SEI's Computer Emergency Readiness Team Resiliency Management Model (CERT-RMM). The company incorporated CERT-RMM using a two-phase process. During the first year, managers applied the model to disaster recovery preparedness planning. In the second year, they expanded the model to include a company-wide analysis of business resiliency.

As a result, Lockheed Martin developed improvement plans for four competencies within its company: people, operations, technology, and process. For example, one of the company's objectives regarding people was to create a skill center to train personnel in the field to handle disaster-recovery situations. One of Lockheed Martin's technology objectives was to become aware of new disaster-recovery and business-continuity threats along with the technologies that could mitigate them.

Mission and risk diagnostics are critical components of CERT-RMM, and they provide an excellent approach to risk management for any company. The mission diagnostic helps organizations to pinpoint the company's mission, the drivers for mission success, the importance of each driver, and how effectively each driver contributes to company success. The risk diagnostic converts mission drivers to risk statements, evaluates how the risk could impact company objectives, and creates mitigation plans. Using these elements together, an organization can create a better business continuity strategy.

Mission Diagnostic

Many current risk management strategies fail to analyze the functions that drive an organization's mission. While the approaches identify potential risks based on current performance, they don't identify the potential risks hidden within the company's vital functions, such as weaknesses in risk management practices. Before trying to pinpoint risks, companies need to know the function, purpose, goals, objectives, or results they want to achieve. Then, they should use that information to articulate a company mission.

When the mission has been identified, it should be broken down into key drivers. SEI recommends asking two main questions: "What circumstances, events, or conditions will create a *successful* outcome?" and "What circumstances, events, or conditions will create a *failed* outcome?" SEI then recommends framing each driver as a yes-or-no question, such as "Is the system used for application deployment sufficient?"

Once evaluators have generated a list of questions, they should gather information and evaluate the questions according to their findings. Each driver should then be weighted according to its importance to the mission. For example, a *critical* driver would be necessary for the organization to function, while a *minimal* driver would have a limited impact on performance.

Once the drivers are weighted, evaluators should return to the mission statement and ask, "How likely is the mission to succeed based on the evaluation of each driver?" Managers of key driver areas can then develop and implement improvement plans.

Risk Diagnostic

Once the mission diagnostic is completed, evaluators should start the risk diagnostic by transforming each driver into a risk statement. For example, "The application deployment process is insufficient" could be a risk statement based on the earlier sample driver.

After creating statements, evaluators should determine how the risks could affect the organization based on the weight of the driver. If a critical driver is not meeting standards, then the company is at great risk. Each risk should then be incorporated into a risk profile. SEI suggests writing each risk on one line of a spreadsheet. Next to the risk, evaluators should rate the risk according to two factors: the probability of exposure and its potential impact.

After developing the risk profile, managers can then choose to mitigate the risk by controlling, watching, or deferring it. *Controlling* means immediate action, *watching* means consistent and frequent reassessment, and *deferring* means putting off action until the next division-wide or company-wide risk assessment.

No risk-management plan will work if stakeholders don't know their roles. Make sure each employee or contractor knows who is accountable for each risk metric, and clearly define each person's role. Supervisors should also be checking the status of improvement plans on a regular basis. Schedule a risk-management review as often as needed depending on the outcome of the diagnostics. Remember, risks may change as new technologies and other threats emerge. Risk is not always bad, and identifying and mitigating risks can help your organization achieve success. **{end}**

Can Anyone Be a Tester?

Testers can come from many different backgrounds, but how does an "anyone can test" mentality affect the craft?

by Julie Hurst | joolery9@gmail.com

Have you heard the rumor going around? Apparently, anyone can be a software tester! Even monkeys can test. This should make our recruitment tasks easy: "You're breathing? You're hired!" Now we have more time for keyboard bashing. Win!

Where has this attitude come from? Is it the arrogance of our coworkers, or could it actually be our own fault? Have some testers paved a negative path for the rest of us to suffer?

There are test teams that sit in the corner, unnoticed and ignored. They do as instructed by non-testers, such as developers and product managers. They don't attend meetings, don't question, and definitely do not challenge statements like "It works on my machine."

Does this happen because of the "anyone can test" mindset, or does the attitude stem from these types of test teams? Perhaps it's a bit of both.

In my first year as a tester, while still learning the ropes, I was led to believe there was no reason for me to learn anything technical. My job was to "play" with the software to find issues and not much more. I easily could've listened, tested as instructed, and found myself stuck in a corner.

> "To spread the news that testers were technically capable and functioning team members, I made a conscious effort to be useful to every team in the company"

However, I was very unsatisfied with this concept of testing and eventually rebelled against it by learning basic programming and using it in my work. What happened next was surprising: My boss discovered that I had developed useful technical skills, moved me from the test team to a team that was more "profitable," and rewarded me with a raise.

Were I more impressionable, I might have taken away from that experience that being a tester involved very little skill or knowledge and that it wasn't the career for motivated individuals like me. But something about testing called to me. When I moved on from that company, I returned to testing and was lucky enough to work with some great people who taught me what testing could really be about.

A few years and jobs later, as a much more experienced and confident software tester, I joined a new company and noticed that the test team was *literally* sitting in the corner, rarely interacting with others outside of the team. Testing was based purely on notes provided by the development team, which were minimal to say the least.

They had no control over their test environments. When I questioned why they required the system administrator to perform application upgrades and configurations, they said, "We don't know how to do it," and, "We have no idea where the files are."

My initial thought was that others in the company must've suppressed these testers, much like in my first testing job. However, this was not true. The developers were incredibly open to more interaction with the testers, but they were so used to working with quiet testers with few technical skills that they did the best they could with them.

Despite an initial positive response to change regarding the test team's involvement in the development lifecycle, it was no simple task to build the respect testing deserved. I couldn't just walk into the room and say, "I'm a tester! I demand respect!" We had to "untrain" these developers from years of dealing with unmotivated testers.

At first, it was an uphill battle to get any decent information out of the developers regarding their work. They answered technical questions with an attitude of "Why would you need to know that?" On a few occasions, an answer they said would be "really technical" or "difficult to understand" turned out to be just a new column in an SQL table or a change in the web.config file.

I made an effort to show off my technical knowledge in the way I worded my questions, defect reports, and test plans. When testers discovered a defect based on technical information shared by developers, I made it known. I infiltrated meetings and discussed the test team's successes with the management team whenever I could.

To spread the news that testers were technically capable and functioning team members, I made a conscious effort to be useful to every team in the company. We tested the company website, stayed late to help the sales team with important demos, and tested (and occasionally even wrote) technical documentation. This showed that we were capable of doing more than just clicking buttons and that, as a test team, we actually

continued on page 37

CollabNet Releases New Version of CloudForge

CollabNet, an enterprise cloud development company, released a new version of CloudForge that offers development tool support, storage availability, and unrestrictive usage of any cloud development platform, including GitHub and Atlassian BitBucket.

The CloudForge Free Plan provides unlimited private projects and supports unlimited users. It also provides access to TeamForge, the enterprise-class agile development platform used to synchronize small teams or thousands of developers and projects. With CloudForge, restrictions are removed so developers and teams can get started instantly, realize value immediately, and securely scale based on their growing and maturing needs.

cloudforge.com

Skytap Inc. Launches Skytap Automation Pack for IBM Rational Team Concert

Skytap Inc., a maker of self-service cloud automation solutions, launched the Skytap Automation Pack for IBM Rational Team Concert. The new product enables IT operations and development teams to rapidly provision IBM build machines on-demand in Skytap Cloud. This advanced integration expands the flexibility of IBM's SmartCloud continuous delivery solution by allowing customers to use scalable, elastic, and on-demand cloud resources to meet sporadic and unpredictable demand for build machine capacity.

The Skytap Automation Pack is software that installs alongside the IBM Jazz Team Server. The Automation Pack continuously monitors the build request queue, and incoming build requests trigger an automated process that creates complete build environments in Skytap Cloud based on pre-defined customer templates. The Skytap Automation Pack enables on-premises IBM Rational infrastructure to connect to Skytap Cloud via a secure VPN connection over IPsec, which is dynamically created using Skytap's AutoNetworks software-defined networking technology.

skytap.com

Klocwork Inc. Unveils Klocwork Cahoots

Klocwork Inc., a software development tool company, unveiled Klocwork Cahoots, a flexible peer code review tool that simplifies the code review process. Klocwork Cahoots fits into the developer workflow to ensure code reviews are both effective and fast. It provides a clean, simple environment for developers to perform code reviews in. Core functions such as starting a review, participating in a review, performing a search, and adding reviewers can each be completed with one click.

Mimicking the interactive nature of social media, Klocwork Cahoots makes it easy to engage in, contribute to, and stay up to date on all elements of active code reviews. Developers can follow individuals, teams, and code components via customizable feeds, monitor activity on an infinite wall, and take part in threaded discussions available on each line of code.

download.klocwork.com

Perforce Software Announces Perforce Commons

Perforce Software announced Perforce Commons, a document collaboration tool that ensures business professionals can more productively work together on files. Commons supports all types of files—from the largest binary objects to the smallest image files—and has merge capabilities for the most common document types, including Microsoft Word and PowerPoint. Because it keeps track of the complete history of any file, it saves business teams significant time and trouble in finding, revising, and collaborating on documents. Document collaboration issues, such as merging multiple edits into a single document or confusion over the most current version of a file, occur as documents pass through many workers and many versions. These issues occur frequently and take a sizeable toll on productivity. Teams of up to twenty users can use Commons completely free of charge. It is also free to all existing users of Perforce Software Version Management products.

perforce.com/product/commons

SmartBear Software Makes CodeReviewer Available

SmartBear Software has made its code review technology available in two new solutions for the SoapUI community as well as for small companies and development teams. CodeReviewer allows development teams with up to ten members to code review at no cost. For teams of up to twenty-five developers, CodeReviewer Pro is available with extensive code review functionality and full technical support at a low cost. Together, both tools join enterprise-ready Collaborator (formerly CodeCollaborator). SmartBear's full product offering allows customers to upgrade when needed. It supports the critical code review practice of organizations of all sizes by focusing on software quality first.

codereviewer.org/downloads/codereviewer-free.html

Compuware Corporation Releases Compuware APM Mobile Application Monitoring Free Edition

Compuware Corporation released the Compuware APM Mobile Application Monitoring Free Edition. With this release, Compuware offers the first free mobile performance management solution for native mobile applications that combines performance, crash, and usage analytics in a unified platform.

Compuware's free mobile solution delivers value across mobile development, testing, and production teams and can be deployed in minutes. The free offering allows organizations to optimize, troubleshoot, and manage business-critical native mobile apps. Customers get immediate insight into end-user responsiveness, crash analysis down to the line of code, and performance analysis by geography, carrier, and more.

compuware.com

SmartBear Software Announces New Version of TestComplete

SmartBear Software, a provider of software quality tools, announced a new version of TestComplete. The new version contains several upgrades, including support for the leading HTML5/JavaScript framework and Sencha Ext JS for rich Internet applications. Additionally, the new TestComplete increases cross-browser testing with support for Opera and Safari as well as updated support for the most recent versions of Chrome and Firefox.

Along with Sencha Ext JS, TestComplete adds support for JavaFX 2, Qt 5, and Apache Flex 4.9. Integration with third-party agile tools is updated to include JIRA 6.0, 5.2, and 5.0 as well as Axosoft OnTime 2012 and 2013. The latest TestComplete works with SmartBear's AQTime 8 application profiler to automatically analyze performance and memory usage and test coverage during test execution.

smartbear.com/products/qa-tools/automated-testing-tools/free-testcomplete-trial

OpsHub, Inc. Releases OpsHub Integration Manager, v5.3

OpsHub, Inc., a provider of agile ALM integration and migration solutions, released OpsHub Integration Manager v5.3, further extending its integration platform with added support for test management assets and artifacts. This release enhances integration between disparate test management and execution tools used by quality teams, improving collaboration and visibility among quality stakeholders. In addition to the existing ALM support, OpsHub Integration Manager v5.3 also adds support for test management assets, further improving support for integration and migration between best-of-breed ALM systems, including HP ALM, Microsoft TFS and VersionOne. OpsHub Integration Manager Version 5.3 is available as a cloud or on-premises solution.

opshub.com

Blueprint Launches Blueprint v5.2

Blueprint, a provider of enterprise requirements definition and management (RDM) software, launched Blueprint v5.2, the latest release of its on-premises and cloud-based solution, which addresses all aspects of the RDM lifecycle. In addition to more than fifty new enhancements and performance improvements, Blueprint v5.2 also features Blueprint for Enterprise Agile, which is a new packaged solution and add-on to the Blueprint RDM software that enables enterprises to improve project results and deliver predictable business value by solving many of the fundamental issues encountered when transitioning from traditional waterfall-style processes to enterprise agile.

In addition to the Blueprint for Enterprise Agile add-on, many other new Blueprint features and performance enhancements are now available in v5.2. These include a fully supported open application program interface (API) that enables third parties to build integrations with Blueprint that provide secure access to the requirements information it holds. This release also supports rich text tables in artifact description fields and rich text properties, which enables users to specify tabular data as part of their requirements.

blueprintsys.com

Reflective Solutions Announces Maximo Pack 3 for StressTester

Reflective Solutions announced the availability of Maximo Pack 3 for StressTester, which ensures that automating Maximo performance testing is now as easy as testing the performance of any other web-based application. Maximo Pack 3 for StressTester includes a selection of sample user journeys for each version of Maximo, in order to help testers to see what these look like and to allow simple performance tests to be executed in a matter of minutes.

reflective.com

QAD Inc. Announces New Release of QAD Enterprise Applications

QAD Inc. announced significant enhancements to the application architecture, functionality, and usability of the latest release of its flagship product, QAD Enterprise Applications. The latest release makes it simpler for QAD customers to maintain, support, and upgrade individual components of their QAD enterprise resource planning (ERP) application, while continuing to leverage their existing system.

QAD Enterprise Applications 2013 provides a powerful new way of delivering upgrades and enhancements with a new modular deployment framework. It allows specific components, elements, and services within QAD Enterprise Applications to be individually enhanced or upgraded. This eliminates disruption from replacements of the entire application. The simplicity of this framework complements QAD's latest implementation methodology, Easy On Boarding, which provides customers with the tools and processes to rapidly deploy or upgrade to a new version of QAD Enterprise Applications either on premises or with QAD On Demand, QAD's cloud ERP solution.

qad.com

Atlassian Releases Beta Availability of Atlassian Connect

Enterprise software company Atlassian announced the beta availability of Atlassian Connect, a new distributed add-on technology that supports third-party customizations and add-ons in its hosted suite of collaboration applications known as Atlassian OnDemand. Atlassian Connect supports standard web protocols and APIs (HTTP, REST), giving developers new choices of programming language and deployment options for creating add-ons for Atlassian applications.

Atlassian Connect will allow developers to remotely host add-ons in isolated containers that can insert content securely into both hosted and, in the future, on-premises installations of JIRA, the industry-leading project management software; Confluence, best-in-class collaboration software; and Atlassian's

developer tools including Stash, a self-hosted Git repository management system. Atlassian's OnDemand customers will be able to customize, integrate, and extend their applications beyond what is currently possible.

atlassian.com

Perforce Software Introduces Perforce Swarm

Perforce Software introduced Perforce Swarm, a powerful code-collaboration and peer-review platform that enables development teams to work together better and faster, improving both the quality of their software and the speed of their delivery. Using Swarm, distributed development teams can improve their agile practices, share creative ideas about their projects, and get early feedback from continuous integration and deployment of their code.

Swarm is a simple web application written in modern web frameworks. It allows developers to personalize their reviews—such as side-by-side or vertical comparisons—and comment right inside the code. Activity streams, project pages, notifications, and a rich repository browser enable even distributed teams to make better use of agile methodologies. Its flexible workflows, hooks for continuous integration, and automatic deployment help teams optimize their software production lines.

perforce.com

Compuware Corporation Announces New Enhancements to Compuware APM for Mainframe

Compuware Corporation announced new enhancements to Compuware APM for Mainframe, an end-to-end transaction management solution spanning the edge of the Internet, through distributed systems, and into mainframe environments. Compuware APM for Mainframe has extended its support for CICS to include CICS Transaction Gateway (CTG) and CICS Web Services (SOAP). These enhancements enable customers who use these technologies to reduce MIPS usage, accelerate mean time to resolution, and deliver better performing applications to their end users.

CTG, SOAP, and Websphere/MQ, which APM for Mainframe already supports, are three of the most common ways to connect distributed applications to the CICS region. These connector technologies have a role in facilitating billions of banking, e-commerce, and insurance transactions that happen each day across the globe. Equipped with the ability to see the impact distributed applications are having on mainframe transactions and workload, enterprises can develop more efficient distributed applications and tune poorly performing ones on the fly.

compuware.com

FAQ

expert answers to frequently asked questions

by Claire Lohr
clohr@computer.org

Will a Tester Certification Help Me Get a Better Job?

The short answer, as in most aspects of the IT industry, is "It depends." There are some fields where certification is a common requirement, such as the PMP for project management and the CISSP for cyber security. Tester certifications have not been around as long and are not as pervasively recognized. They help, but how much depends on the context of the job search.

Maximum Results

Companies that are international or about to become international: As of September 2012, ISTQB has issued more than 265,000 certifications in more than seventy countries worldwide, with a current growth rate of approximately 10,000 new certifications being issued per quarter. In the US, as of the time of this writing, there are 13,073—a much smaller percentage of the eligible workforce. In other countries (particularly Europe, where this certification started), employers are much more concerned about applicants' having "hard" (as opposed to "soft," or opinion based) credentials that come from an external source.

Organizations that are certifying all testers: There are a number of large employers who are certifying all testers or anyone who works closely with testers as a part of an effort to improve. They will be delighted to get someone who is already certified and has the common vocabulary and knowledge of best practices.

Medium Results

Older workers: The concern of a hiring manager is that a worker who touts thirty years of experience may actually have one year's experience thirty times. A certification validates the currency of the job seeker's knowledge.

Those entering this field or workers who have been out of the field for a while: Again, a certification validates the job seeker's knowledge. What's more, ISTQB certification has no recertification requirement and lasts forever.

Applying for a job within your company: Even if the hiring manager has not heard of the certification, it can be a tiebreaker when there is another equally qualified candidate.

Applying for a job at a new company: Certifications may not help you get the interview—that typically is based on meeting a company's established list of requirements—but the interviewer will ask what those initials after your name represent and may even be impressed when you explain it.

Pervasive Benefits

More confidence: Experienced testers have more confidence that they are doing it the right way and are not missing pertinent knowledge.

Use of industry-recognized terminology: Many firms—or even different projects within one organization—use their own nonstandard terms. It is much better to describe work experience in a résumé using terms that are recognized by the industry.

Learning something new: Almost everyone, especially novice testers, learns something new that will enable her to do a better job with the same resources.

Just Do It

The benefits of greater confidence while using better techniques and vocabulary are immediate, and being prepared for a potential planned or unplanned change of employment is best done before the need arises. **{end}**

continued from page 32

knew the software better than anyone else did.

Eventually, the test team got the respect we deserved from the entire company. We were no longer forgotten, were no longer in the corner, and were seen as an integral part of the process. It was a great feeling. While I could easily blame the previous test teams for all the time and effort needed to build respect for my team and me, the end result was well worth it. However, this was only one small step toward rebuilding the damage that other testers have caused with their nonchalant attitudes.

We can sit back and let this attitude continue to exist in our industry, or we can start giving future software testers more opportunity to shine and thrive. While many of us may not have planned our careers in software testing, we have chosen to stick with it and therefore need to stand proudly, take responsibility for the craft, and accept that it is our responsibility to get the respect we deserve. **{end}**

index to advertisers

ADC/BSC East 2013	adc-bsc-east.techwell.com	Back Cover
ASTQB	www.astqb.org	35
Cognizant	www.cognizant.com/testing-services	25
Cygnet	www.testing-whiz.com	18
OpenMake Software	www.openmakesoftware.com	19
STAR*WEST* 2013	starwest.techwell.com	9, 11, 13, 15
SQE Training	sqetraining.com/trainingweek	Inside Front Cover
Telerik	www.telerik.com/html5-testing	2

Display Advertising
advertisingsales@sqe.com

All Other Inquiries
info@bettersoftware.com

Better Software (ISSN: 1553-1929) is published six times per year: January/February, March/April, May/June, July/August, September/October, and November/December. Back issues may be purchased for $15 per issue plus shipping (subject to availability. Volume discounts available.

Revolutionize Your Software

BOSTON, MA • NOVEMBER 10–15, 2013

Super Early Bird Savings!

REGISTER BY SEPT. 13 AND
SAVE UP TO $400
GROUPS OF 3+ SAVE EVEN MORE!

Two Conferences in One Location

Register and attend sessions from both events!

EXPLORE THE FULL PROGRAM AT

adc-bsc-east.techwell.com

PMI® members can earn PDUs at this event

PMI R.E.P. Project Management Institute