

Summer 2016

www.TechWell.com

BETTERTM SOFTWARE

A TECHWELLTM PUBLICATION

#WOMENINAGILE
Perceptions of
Women in Agile

PRODUCT-DRIVEN PROCESS
Putting the Product First

BUILDING A SOLID FOUNDATION FOR YOUR DEVOPS TRANSFORMATION



Over 100+ Learning Sessions in 1 Location!

Agile Dev Better Software DevOps **EAST**

A TECHWELL EVENT

November 13–18, 2016

Orlando, FL | Hilton Orlando Lake Buena Vista

**CHOOSE FROM A FULL WEEK OF LEARNING,
NETWORKING, AND MORE**

SUNDAY Multi-Day Training Classes Begin

MONDAY-TUESDAY In-Depth Half- and Full-Day Tutorials

WEDNESDAY-THURSDAY Keynotes, Concurrent Sessions,
the Expo, Networking Events, and More

FRIDAY Agile Leadership Summit

Special Offer for *Better Software*
Subscribers: Register using promo code
BSMCE16 by Sept. 16, 2016 to save up to
an additional \$400 off your conference*

bsceast.techwell.com

*Discount valid on
packages over \$400

Keynotes by International Experts

Agile Dev
Better Software
DevOps **EAST**
A TECHWELL EVENT



Lead Teams that Deliver the Goods

Andy Kaufman
Institute for Leadership Excellence and Development, Inc.



Building Product Development Communities: From Startups to the Enterprise

David Hussman
DevJam



Solve Everyday IT Problems with DevOps

Sherry Chang
Intel



Agile Metrics: Make Better Decisions with Data

Larry Maccherone
AgileCraft

JUST A FEW OF OUR IN-DEPTH HALF- AND FULL-DAY TUTORIALS

IoT Testing Workshop

Jennifer Bonine, tap|QA

Writing Agile User Story and Acceptance Test Requirements

Robin Goldsmith, GoPro Management

Creating a Continuous Delivery Pipeline: A Hands-On Workshop

Ken Mugrage, ThoughtWorks

Fostering Sustained Agility

Tricia Broderick, Agile For All

Continuous Delivery: Rapid and Reliable Releases with DevOps

Bob Aiello, CM Best Practices Consulting

The Tester's (New) Role in Agile Development

Rob Sabourin, AmiBug.com



NOVEMBER 16-17 **THE EXPO**

Discover the Top Technologies and Tools All Under One Roof!

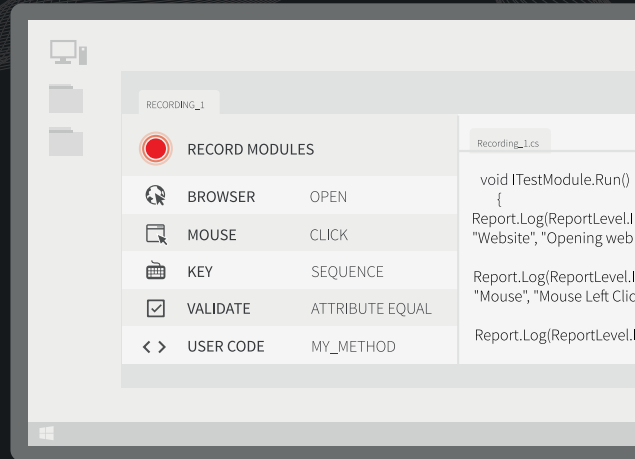
**TOOLS
SERVICES
TECHNIQUES
DEMOS**






WHO SHOULD ATTEND?

Software managers, directors, CTOs, and CIOs, project managers and leads, measurement and process, improvement specialists, requirements and business analysts, software architects, security engineers, test and QA managers, developers and engineers, technical project leaders, testers, process improvement staff, auditors, business managers

TO REGISTER CALL 888.268.8770
bsceast.techwell.com

Test Automation for Everyone



-  Any Technology
-  Seamless Integration
-  Broad Acceptance
-  Robust Automation
-  Quick ROI

1 License
All Technologies.
All Updates.

www.ranorex.com/try-now



CONTENTS



in every issue

- Mark Your Calendar **4**
- Editor's Note **5**
- Contributors **6**
- Interview with an Expert **10**
- Ad Index **37**

Better Software magazine brings you the hands-on, knowledge-building information you need to run smarter projects and deliver better products that win in the marketplace and positively affect the bottom line. Subscribe today at BetterSoftware.com or call 904.278.0524.

features

12 COVER STORY **BUILDING A SOLID FOUNDATION FOR YOUR DEVOPS TRANSFORMATION**

To deliver quality software with speed and stability, a huge shift is needed in the way technology is managed throughout any organization. Nicole Forsgren believes that establishing the right culture is vital, especially during DevOps adoption.
by Nicole Forsgren

18 PITFALLS OF DEVELOPING FOR THE IOT

The Internet of Things (IoT) enables amazing software-powered devices designed to make our business and personal lives easier. Lev Lesokhin discusses four fundamental practices you'll need when developing sophisticated software for the IoT.
by Lev Lesokhin

24 RECOGNIZING #WOMENINAGILE

As part of her involvement with #WomeinInAgile, Natalie Warnert conducted a study to determine why women are less involved in the agile community and what can be done about it. Her research shows some surprising results.
by Natalie Warnert

30 MAKING THE MOVE TO PRODUCT-DRIVEN PROCESS

Just because you follow the rules of your software development process doesn't necessarily guarantee project success. According to David Hussman, there are four product-centered principles that everyone should practice.
by David Hussman

columns

7 TECHNICALLY SPEAKING **GETTING TEST TO THE STRATEGY TABLE**

When the big decisions are made to fund and scope the project, are testers in the room? Matt Heusser presents compelling reasons for including QA as a key contributor during strategic planning, along with advice on how to get there.
by Matt Heusser

36 CAREER DEVELOPMENT **CRACKING THE CODE ON MILLENNIALS**

Our latest generation of programmers, project managers, and testers is perceived to be uninterested, unmotivated, and difficult to manage. Jason Garber presents innovative techniques you can use to lead your next rising star.
by Jason Garber

MARK YOUR CALENDAR

training weeks

Testing Training Week

<http://www.sqetraining.com/trainingweek>

August 22–26, 2016

Dallas, TX

September 19–23, 2016

Washington, DC

October 17–21, 2016

Tampa, FL

November 7–11, 2016

San Francisco, CA

software tester certification

<http://www.sqetraining.com/certification>

August 16–18, 2016

Philadelphia, PA

August 22–24, 2016

Dallas, TX

September 13–15, 2016

Seattle, WA

September 19–21, 2016

Washington, DC

September 27–29, 2016

Boston, MA

October 17–19, 2016

Tampa, FL

October 24–26, 2016

Chicago, IL

November 1–3, 2016

Raleigh, NC

conferences

STARWEST

<https://starwest.techwell.com>

October 2–7, 2016

Anaheim, CA

Disneyland Hotel

STARCANADA

<https://starcanada.techwell.com>

October 23–28, 2016

Toronto, ON

Hyatt Regency Toronto

STAREAST

<https://stareast.techwell.com>

May 7–12, 2017

Orlando, FL

Rosen Centre Hotel

Agile Dev East

<https://adceast.techwell.com>

November 13–18, 2016

Orlando, FL

Hilton Orlando Lake Buena Vista

Better Software East

<https://bsceast.techwell.com>

November 13–18, 2016

Orlando, FL

Hilton Orlando Lake Buena Vista

DevOps East

<https://devopseast.techwell.com>

November 13–18, 2016

Orlando, FL

Hilton Orlando Lake Buena Vista



SOFTWARE DEVELOPMENT AND CULTURE

In our feature cover article, "Building a Solid Foundation for Your DevOps Transformation," Nicole Forsgren shows how culture influences the success of any organization's transformation to embrace DevOps—continuous automation of the development, validation, and deployment of enterprise software.

With the millions—or should I say billions?—of possible Internet of Things devices becoming a part of every household and business, Lev Lesokhin presents some lessons learned with his insightful "Pitfalls of Developing for the IoT." Even though IoT targets tend to be small, singular-focused devices, the effort required to develop and test embedded systems is far from simple.

David Hussman's "Making the Move to Product-Driven Process" makes the case to put the customer, learning, and product front and center rather than focusing on the mechanics defined by the agile methodology used. With years of experience coaching software development teams, David says the cultural shift to build products aimed at delighting the customer is a necessity.

And speaking of culture, Natalie Warnert researched how men and women perceive female involvement in the agile community for her timely article "Recognizing #WomenInAgile." If you aren't aware of the movements promoting women in our industry—including Women in Agile, Girls in Tech, Girls Who Code, and Women Who Test—Natalie's article will enlighten you about the campaigns working to close the gender gap in the tech world.

For those of you working in a testing organization and wishing that your peers and management listened more to your concerns about product quality, Matt Heusser presents some great advice in his article "Getting Test to the Strategy Table." And just when you think you've figured out Millennials in the workplace, Jason Garber summarizes the benefits that the youngest working generation brings to software development with "Cracking the Code on Millennials." All it takes is the right leadership.

We truly value your feedback. Let me know what you think of the articles and the new cover art created just for us by Mike Varouhas by leaving me your comments. I sincerely hope you enjoy this issue and please use social media to spread the word about *Better Software* magazine. THANK YOU!

Ken Whitaker
kwhitaker@techwell.com
Twitter: @Software_Maniac

Publisher
TechWell Corporation

President/CEO
Wayne Middleton

Director of Publishing
Heather Shanholtzer

Editorial

Better Software Editor
Ken Whitaker

Online Editors
Josiah Renaudin
Beth Romanik

Production Coordinator
Donna Handforth

Design

Creative Director
Catherine J. Clinger

Advertising

Sales Consultants
Daryll Paiva
Kim Trott

Production Coordinator
Alex Dinney

Marketing

Marketing Manager
Cristy Bird

Marketing Coordinator
Kelly Radell

Cover art © Michael Varouhas

FOLLOW US



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
TechWell Corporation
350 Corporate Way, Suite 400
Orange Park, FL 32073

Contributors



NICOLE FORSGREN is an IT impacts expert who shows leaders how to unlock the potential of technological change in their organizations. Best known for her work with tech professionals and as the lead investigator on the “State of DevOps Reports,” Nicole is a consultant, expert, and researcher in knowledge management, DevOps, IT adoption, and impacts. She is the director of organizational performance and analytics at Chef Software and an academic partner at Clemson University. Contact Nicole at nicolefv@gmail.com.



JASON GARBER is a senior software engineer and cofounder of PromptWorks, a Philadelphia-based consulting shop that delivers customized web and mobile web applications. Jason founded his first website development company at age fourteen and later became a passionate advocate for Ruby on Rails, clean code, and automated testing. He has consulted with clients through his web development work and previously held positions at Rosetta Stone and Eastern Mennonite University. Contact Jason at jason@promptworks.com.



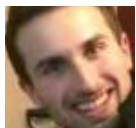
The managing director at Excelon Development, **MATTHEW HEUSSER** is probably best known for his writing. In addition to serving as technical editor of StickyMinds.com, Matt was the lead editor for *How to Reduce the Cost of Software Testing*. He has served as a board member for the Association for Software Testing and as a part-time instructor in information systems for Calvin College. Reach Matt at matt@xndev.com.



As leader of DevJam, a company composed of agile collaborators, **DAVID HUSSMAN** works with companies of all sizes worldwide as a teacher and coach on the adoption of agile methods as powerful delivery tools. David often works with leadership groups to pragmatically use agile methods to foster innovation and a competitive business advantage. Previously, he spent years building software in the audio, biometrics, medical, financial, retail, and education sectors. Contact David at david.hussman@devjam.com.



Executive vice president of strategy and analytics at CAST, **LEV LESOKHIN** is responsible for market development, strategy, thought leadership, and product marketing worldwide. He has a passion for making customers successful, building the ecosystem, and advancing the state of the art in business technology. Lev previously held positions at SAP, the Corporate Executive Board, McKinsey & Company, and MITRE Corporation. Reach Lev at l.lesokhin@castsoftware.com.



A longtime freelancer in the tech industry, **JOSIAH RENAUDIN** is now a web content producer and writer for TechWell, StickyMinds, and *Better Software* magazine. He also writes reviews, interviews, and long-form features for popular video game journalism websites like GameSpot, IGN, and Paste Magazine. Josiah has been immersed in games since he was young, but more than anything, he enjoys covering the tech industry at large. Contact Josiah at jrenaudin@techwell.com.



NATALIE WARNERT is a recognized thought leader in the agile community and a repeat author for *Better Software*. Her passion for product development and agile principles radiates through her writing and speaking engagement topics. Natalie is also active in the #WomenInAgile community and strives to increase involvement and diversity in the agile industry. Find out more at www.nataliewarnert.com.

Getting Test to the Strategy Table

Often we preach the importance of quality throughout the software development process. Quality is more strategic than you might think.

by **Matt Heusser** | matt@xndev.com

If you've been paying close attention to the agile-scaling literature lately, you may have noticed two trends. First, that our community seems to recognize the need of the technical architect, perhaps CTO, at the highest levels of at least IT, if not the entire company. Second, testing isn't so lucky.

In fact, it seems to be going the other way. If you look at the descriptions of the tester in the Scaled Agile Framework or Disciplined Agile Delivery, the job of the tester involves creating automated tests that generate a green bar when they pass and monitoring the health of those tests over time. The term testing is never really defined well. If we're lucky, we might see a reference to the test automation pyramid, or perhaps a reference to exploratory testing.

Meanwhile, as James Whittaker said five years ago, it is the tester in the back of the room, watching the video on the company's future products, who is able to say "this stuff is never gonna work." [1] And by the way, those future products James talked about *still* don't exist.

The industry is pushing testing into a replaceable, commodity role that offers little value. In fact, testers actually have the insight to help the company make good decisions about what to build.

In other words, testers want a seat at the strategy table and they have something to contribute.

Regardless of a company's style and size, I have identified a few patterns worth repeating.

Reality about the Strategy Table

First, allow me to deliver some bad news. As Neo discovers in the movie *The Matrix*, there is no spoon. For testers, there is no strategy table.

That is, the places you go to—the meetings that feel like things are discussed—are often more like *theatre*. The decisions have already been made. The activity that is actually happening in that meeting is the signing of the bill into law, and the roll

call has already been taken. Testers tend to misunderstand this. We go to a meeting where the decision has already been made and blurt out information that people wish they had heard yesterday. At the same time, we can embarrass the people who wanted the decision made their way.

In my experience, the bad decision gets made anyway and after our intervention, we've made an enemy to boot. If we manage to get the decision undone, well, good for us! We've just made an enemy for life.

Early in my career, I was participating in a steering committee meeting to discuss a company's ERP system. I was in the room with vice presidents and directors. After much discussion, I politely explained that the strategy of real-time processing they wanted to pursue wasn't possible with their just-licensed multimillion-dollar ERP system.

Everyone was stunned and left the meeting scratching their heads. I was uninvited from the group. At the next meeting, without my meddling, the group decided that they did not need to pursue real-time processing at that time, and I got a note on my annual review that I "could be difficult to work with."

Looking back, I made my director, the technology group, and everyone else on the steering committee backing the idea look bad. If

I had waited and talked to my director privately, we could have found a way to broach the subject without losing face. John P. Kotter, the Harvard professor and best-selling author, calls this process *building a coalition*—step two in his change management model. [2]

Building a coalition allows us to do something else: create a strategy table. By figuring out who has influence and providing them with good information, we can become influential ourselves.

The second great mistake testers make is confusing strategy with saying no. What we actually did was make ourselves appear to be negative and not helpful, resulting in upset and no clear solution.

“The industry is pushing testing into a replaceable, commodity role that offers little value. Testers actually have the insight to help the company make good decisions about what to build.”

Adding Value to Strategy

If this whole article were about IT strategy, it would still be too broad a scope. For now, let us say that strategy is about how the organization will accomplish high-level goals, how it creates those goals, and what those goals are. They could be company, division, or IT-level goals. It doesn't really matter.

The problem is that executives, especially founders, are used to doing unreasonable things. They ignored all the advice to play it safe and have done spectacularly well. Even worse, they found that a large percentage of the time it works to say "Figure it out!"—or at least it *appears* to work.

So here's one idea.

A few years ago, I worked on a software product that was not supposed to support tablets. Yet tablets were being used by customers anyway, resulting in millions of dollars of revenue. We also had some pretty big known issues and some bigger known usability problems. Management called the dream team together and asked, "What are you going to do about it?"

The room was heavy with testers, who had been told to ignore the issue, to not worry about it, until suddenly we were in trouble for it. We could have used that as an excuse and gotten out of the room.

But I *wanted* to be in that room.

Instead of complaining, I suggested that we lock me in a room for three days to do serious exploration of the product and to figure out how big the tablet support issues were. Once we knew the size of the problem, we could figure out mitigation and public relations strategies, which the testing team was front and center on. The effort succeeded. The short list of must-fix bugs was indeed short, but the potential feature list was much larger. That led into a tablet project, where testing slowly became more involved throughout the entire product development process.

So find your opportunity and take it. I admit, it is unlikely that you'll see a chief testing officer at your company anytime soon. Still, you might help create a culture where testing is more than the people that make the tests go green.

And that is a very good thing. `{end}`

Sticky
Notes

Click here to read more at StickyMinds.com.

■ References

CAN'T ATTEND A TECHWELL CONFERENCE? NO TIME FOR FORMAL TRAINING? WE'VE GOT YOU COVERED.

Check out the **TechWell Happenings YouTube Playlists**.

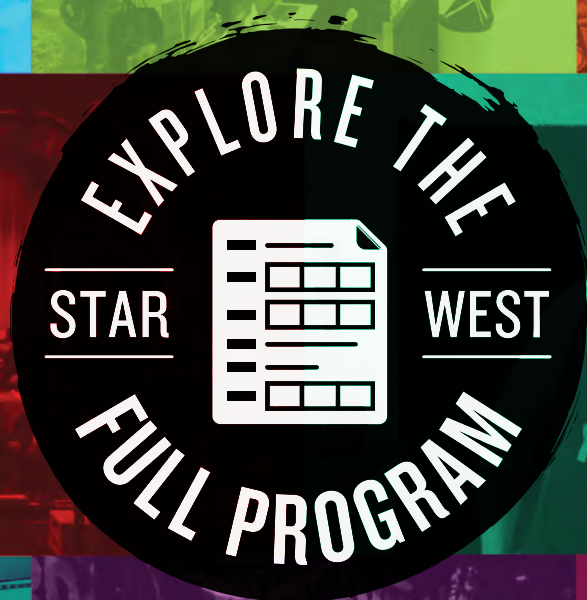
Hundreds of interviews, lightning talks, and STAREAST, STARWEST, and Better Software conference presentations are grouped by topic, so it's simple to take control of your learning experience.

Covering software testing and development topics ranging from mobile testing to enterprise-level agile development and pretty much everything in between, **TechWell Happenings Playlists** deliver expert-level knowledge directly to you, for free, whenever you want it.

Visit **well.tc/TWHapps**, to subscribe to the **TechWell Happenings Channel** so you won't miss out on the newest interviews and TechWell conference presentations.

Be a Part of the Action at **STARWEST 2016**

Register by August 5 and Save Up to \$400 off!*



October 2-7, 2016

Anaheim, CA | Disneyland Hotel

STAR WEST

A TECHWELL EVENT

[HTTPS://WELL.TC/STARWESTBSM](https://well.tc/starwestbsm)

*Discount valid on packages over \$400

Anders Wallgren

Years in Industry: **27**

Email: wallgren@electric-cloud.com

Interviewed by: **Josiah Renaudin**

Email: jrenaudin@techwell.com

“I think the first ‘Oh, crap!’ moment that everybody has when they start thinking about DevOps and continuous delivery and continuous deployment, and all of those kinds of things is: ‘What do you mean you’re going to check code fifteen minutes into production?’”

“As organizations started to adopt agile and get better at delivering the software, at least out of the development and QA organizations, the logical next question was, ‘Well, how do we get it into customers’ hands more quickly?’”

“I think you have to start small. If agile has taught us anything, it’s ‘divide and conquer’: Break ethics down into stories, break stories down into smaller things that you can work on. Don’t have eighteen-month releases; have two-week cadences.”

“If you speak to pure agilists, they’ll happily inform you that the Agile Manifesto always talked about delivering software to customers, which is a true statement, but I think the practical applications of agile were very often felt much more on the development side of the house than the ops side.”

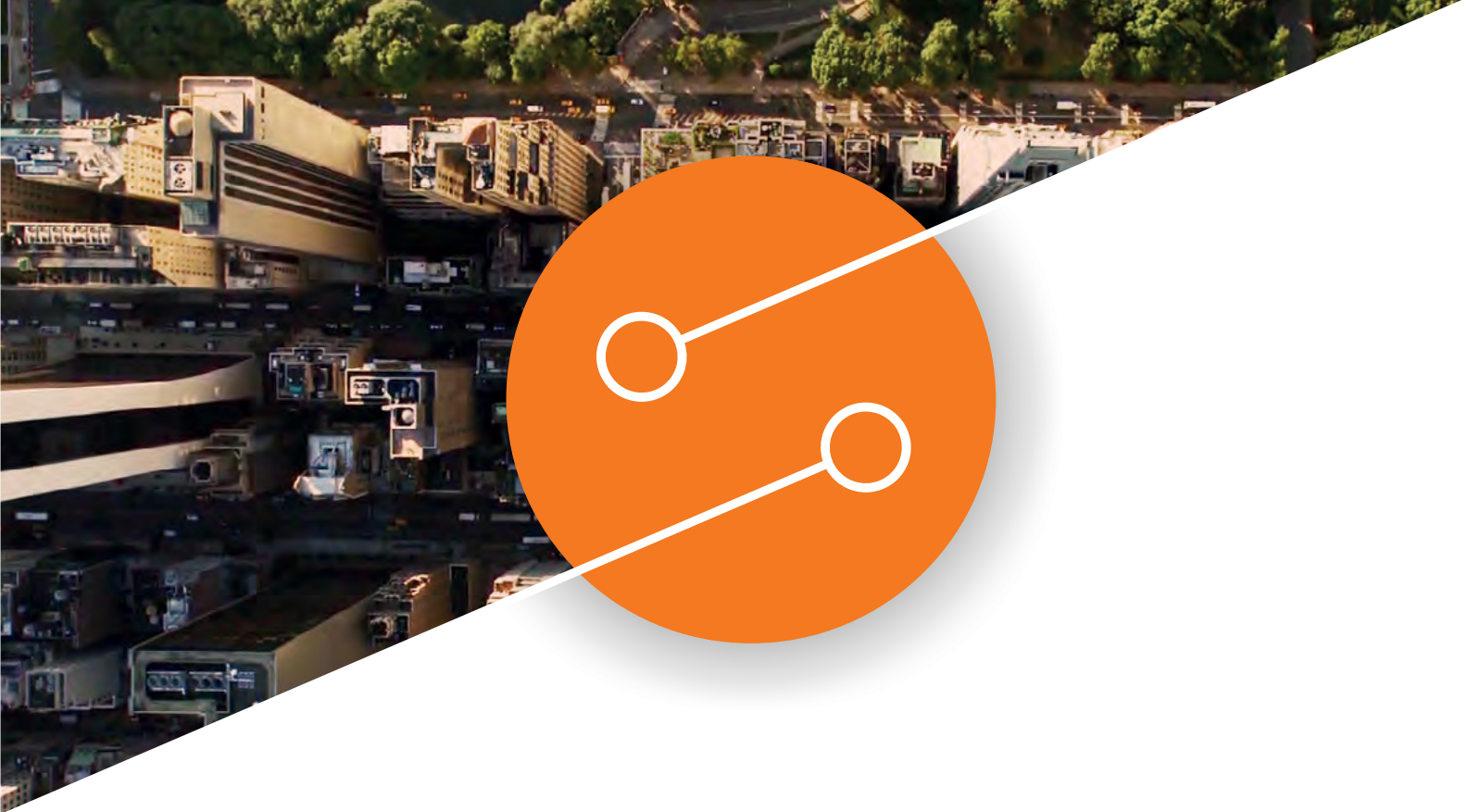
“We’re going to have to mess up some organizational chemistry in the way that ‘You can’t make an omelet without breaking a few eggs,’ but all in the name of getting everybody to work better together, not just for the sake of messing it up.”

“The record shows again and again that a well-crafted, well-maintained, and high-performing DevOps team exposes itself and its customers to fewer risks, not more.”

“There are people who are betting their careers on DevOps transformations, and that’s a big deal; you may not have your job in a year or two if it doesn’t go well.”

“You need to, as an organization, understand from laptop to live, or room to tune—or whatever sort of analogy you want to use—how your code travels from the time that it’s submitted to the source code control system to when it’s live on the site.”

For the full interview, visit <https://well.tc/IWAE18-3>



InfoStretch helps enterprises accelerate time-to-market of their mobile, digital, and IoT initiatives.

Areas of InfoStretch support and expertise include:

- Strategy & Blueprinting
- Application Development
- Quality Engineering
- Test Automation
- Continuous Integration & Continuous Delivery

Request a Free Consultation Today at
www.infostretch.com/getstarted

www.infostretch.com

infostretch
Accelerating the mobility of things



BUILDING A SOLID FOUNDATION FOR YOUR DEVOPS TRANSFORMATION

BY NICOLE FORSGREN

With technology transformations driving innovation in every industry, and DevOps lighting up conversations from engineers to the executive suite, the days of thinking IT doesn't matter are decidedly gone. [1] This new wave of doing business is shaking up the world because it touches every industry vertical—retail, banking, finance, government, and, of course, technology. Almost every company today is a software company, and every software company is in a race to keep up with the demand for innovation.

The Emergence of DevOps

Today's marketplace demands a crushing pace for those who want to not merely survive but to win. Never before has there been such pressure to deliver quality software solutions so quickly. Customers expect innovation and excellence, and if they don't find it from you, they will move on. Competition is fierce with new businesses entering the market regularly and existing competitors continually improving their own products to keep pace.

Software is at risk in ways that manufacturing and retail never were. Security threats and vulnerabilities from faraway actors require fast responses and updates. Legal and regulatory policies may require updates to software and IT infrastructure. The ability to deliver quality software with speed requires a huge shift in the way technology is managed throughout any organization. DevOps is making this transformation possible.

DevOps is a revolutionary change in the way technology is developed and software is delivered. In contrast to the model where technology has been a separate component that was purchased or outsourced and then plugged into an organization, DevOps is a full value-chain activity that involves a number of roles, including a business champion, engineers, testers, and IT operations. These cross-functional teams allow rapid development that delivers business value, but DevOps also requires drastic changes to the way work is done. This shift is similar to the changes seen in lean and Toyota manufacturing methods, emphasizing not only technology and tooling but also process and culture.

Technology, Process, and Culture

Too often, I see organizations implement their own tech transformation in the prescribed order of technology, process, and then culture. They focus on technology first because it's the shiny new toy to play with or because it's the easy problem to fix (just write a check). Update the tech stack (LAMP or ELK), use the hot new thing (containers), and we're bound to be successful, right?

Organizations usually set their sights on improving the process. They organize key players to implement new processes: Do A, then B, then C. They roll out agile and have stand-up meetings instead of the traditional, long-winded sit-down meetings. If they're lucky, they get around to culture.

But this ignores the truism that DevOps experts and practitioners have been preaching for years: You must also have the right culture. Without a supportive culture, the technology starts to fail, and the process will collapse. Just as the classic

Roman arch crumbles without the keystone at the top, technological changes—especially those as ambitious and transformative as DevOps—falter and fail without a solid, supportive culture with which to anchor the work.

Many cite Etsy as a prime example of what DevOps culture should look like. According to Etsy's engineering management, getting the culture right was key to the company's IT transformation from good to great. [2] Amazon is another company known for DevOps excellence. Greg Linden, the developer who designed the shopping cart recommender, said, "I think building this culture is the key to innovation. Creativity must flow from everywhere.

Whether you are a summer intern or the CTO, any good idea must be able to seek an objective test, preferably a test that exposes the idea to real customers. Everyone must be able to experiment, learn, and iterate." [3]

Is Culture Really Transformative?

Stories are powerful, and anecdotes about the importance of culture abound. We can see in the data the critical role that culture plays in DevOps transformations. I have studied the effects of cultural and environmental factors on technology for almost a decade, and for the past three years, I have been the lead investigator on the Puppet Labs annual State of DevOps Report. [4, 5, 6]

These are the largest studies of DevOps practices and impacts to date, collecting more than twenty-five thousand survey responses from DevOps professionals representing more than a thousand organizations in hundreds of countries and dozens of industry verticals. The data show that culture has an impact not only on IT performance but also on overall organizational performance. So, what do we mean by *culture*? Basically, culture is what it feels like to work in your company and on your teams. It includes your thoughts and emotions, and it influences the actions you take. It lets you know what you can expect from your colleagues.

In the State of DevOps studies, we developed an assessment of culture based on research done by Ron Westrum, who found that cultures high in trust and information flow are predictive of performance outcomes, particularly in complex, high-risk work environments like health care and aviation. Westrum's work is especially interesting because it tells us what happens when things go wrong. [7] As technology professionals, we know that can happen quite often.

Although Westrum's cultural framework hadn't been tested in DevOps or technology environments, we knew software development was complex and high-risk, prone to errors and failures. True to our hypotheses, we found similar patterns that cultures fostering information flow and trust drive performance. Westrum classifies organizational culture into generative, bureaucratic, and pathological groups. In our own study, 33 percent of survey respondents reported working in a generative (high-trust) culture. This group had the highest IT performance outcomes.

Only 15 percent of survey respondents reported working in a pathological (command-and-control) culture, and they saw the lowest IT performance outcomes. By far the largest number,

52 percent, of respondents reported working in the middle environment, the bureaucratic (rule-oriented) culture, which saw moderate IT performance outcomes.

Table 1 shows the importance of cooperation, sharing risks, failure leading to learning, and embracing novelty among the different culture groups. The results from our State of DevOps studies lined up with Westrum’s framework of how culture influences performance outcomes. Where do your team and organization fall?

How Culture Impacts Team Effectiveness

An analysis from Google’s People Operations team (the mega company’s version of HR) highlights the importance of culture in team effectiveness. [8] When the research team initially started interviewing employees, they expected to find some magical mix of individual traits and skills necessary for assembling a dream team.

What the tech giant found instead was that high-performing teams are driven less by who is on the team and more by a team culture that fosters communication, structure, and meaningfulness in the work. And the most important aspect of this team culture? Psychological safety. This essential ingredient refers to the trust between members to take risks in their work and be vulnerable around each other. Google’s thoughts on culture are similar to the generative culture in table 1.

Although many in tech may be surprised that culture is that important and teams matter more than individuals, Toyota knew it years ago, based on its manufacturing process that focuses on continuous improvement and efficiency. Toyota’s success was enabled by fostering a culture of trust, teamwork, and improvement through all levels of the organization. Perhaps the clearest example of the effect of culture on performance

from Toyota lies in the story of the NUMMI automotive plant.

In 1984, GM and Toyota opened a joint venture called New United Motor Manufacturing Inc. (NUMMI) on the site of the former General Motors Fremont assembly plant. Historically, this manufacturing plant was infamous for poor quality and assembly line workers who exhibited bad behavior while on shift. As a part of this new joint venture, Toyota was convinced to hire most of the prior GM workforce—the same workforce that did such abysmal work.

Several of these workers were sent to Toyota City in Japan to learn the Toyota Production System, and they returned to NUMMI armed with a new way of working and immersed in the Toyota work culture. On site, they passed on their new knowledge—of both the production system and the culture—to their colleagues. Within months, the first car rolled off the production line, and the NUMMI plant was producing cars with the same quality and low defect rates as those seen in Japan.

It was the same workers, just operating within a different culture. What caused such a change in the NUMMI workers? Toyota’s manufacturing process made it safe to ask for help when things went wrong. If any work could not be completed in an allotted amount of time, the worker rang a bell, which resulted in a manager coming to help solve the problem. If another specified amount of time passed and the problem was still not resolved, the andon cord was pulled, stopping the entire production line.

During this time, key players swarmed around the problem to troubleshoot. What could improve the work? Could a tool’s design be modified? Could a process somewhere else along the line be changed? The shared vision and collaboration at Toyota were so complete that even upstream suppliers were willing to modify the design of their parts to improve the process.

Pathological <i>Power-oriented</i>	Bureaucratic <i>Rule-oriented</i>	Generative <i>Performance-oriented</i>
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

Table 1: The three groups of organizational culture (according to Westrum)

The Right Culture Benefits Performance

The Toyota example shows us a culture of trust, teamwork, and improvement. Trust—that asking for help will be met with assistance, not with blame or punishment for failure to complete a task on time. Teamwork—centered around solving problems together with key players within the company. Improvement—of the parts and process whenever the work deviates from the plan. And beyond this, a culture of continued improvement, because all team members knew the end goal was producing a quality car. In fact, Toyota welcomed visitors to its production plants and had no concerns when they took copious notes about Toyota's processes. Because it wasn't the process that brought a competitive advantage; it was the culture. This is the power of culture with performance.

Of course, technology is important as well. Effecting rapid change is easier on a greenfield project, starting out with new technology and without the constraints of prior work, than on legacy systems. But don't let an old technology hold you back if that is where you find yourself. A trusting culture with good information flow that prioritizes safety in teams can make even legacy or hard technology an innovative and exciting place to be.

A great example is the work of Gary Gruver and his colleagues who worked at Hewlett-Packard on the HP LaserJet. [9]

The team was able to implement DevOps in firmware and see marked increases in efficiency, freeing up time and giving the team eight times the capacity for innovation. This transformation relied on a culture change that prioritized quality collaboration. Similar to the Toyota way, the team stopped all work when something broke a build. This approach enabled collaboration and information flow to solve problems early.

Make a Cultural Move in Your Organization

What can you do? Let's reference the Westrum framework for some clues. As an IT leader, you can create and support cross-functional teams to support high cooperation. As a team member, you can do your part to reach out to your peers, both inside your own team and across teams.

Conducting blameless postmortems following an outage or failure is also a highly cited best practice in the DevOps community, and for good reason. This positive style of handling postmortems supports two important areas of the Westrum framework: training the messenger and failure leading to inquiry.

Technology transformations are undertaken because of the exciting rewards they can offer, but all change introduces risk. As organizations and teams transform, these risks and responsibilities should be shared. This implies that all of us should take ownership of things like quality, availability, reliability, and security. Start conversations about aspects of technical work that are overarching and that everyone can contribute to. When Westrum encourages bridging, this is the essence of DevOps. In other words, look for any excuse to break down silos in your work.

Finally, implement novelty by encouraging experimentation and making it safe to fail on new ideas that could bring value

to the work.

Armed with these tools and techniques, you can improve and support your own culture. With an open, collaborative culture providing a solid foundation for your tech transformation, you can drive innovation and provide real value to your business and your customers. {end}

nicolefv@gmail.com

Sticky
Notes

[Click here](#) to read more at StickyMinds.com.

■ References

GET INSPIRED *in* TORONTO

WITH THESE KEYNOTES



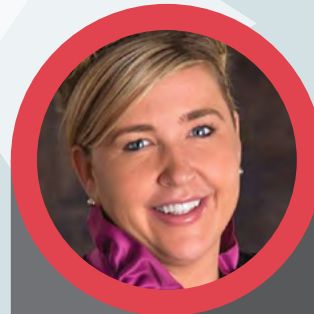
Testers in Agile Teams—Isolation or Collaboration?

Rob Sabourin, AmiBug.Com



Evolution—Not Revolution: Transforming Your Testing

Julie Gardiner, Hitachi Consulting



Testing Leaders in the Digital Age: Are You Adapting?

Jennifer Bonine, tap|QA, Inc



The Future of Applications and Application Testing

Dwayne Forde, Pivotal



SAVE UP TO \$300 WHEN YOU REGISTER BY AUGUST 26

STAR CANADA

A TECHWELL EVENT

**October 23–28, 2016
Toronto, ON**

Hyatt Regency Toronto

Learn More: <https://well.tc/starcanadabsm>

FULL- AND HALF-DAY TUTORIALS INCLUDING:



Selenium Test Automation: From the Ground Up

*Jeff "Cheezy" Morgan,
LeanDog*



Test Data: Create, Secure, Manage, and Reuse

*Chip Groder, Intervise
Consultants Inc.*



Measurement and Metrics for Managers

Mike Sowers, TechWell Corp.



Critical Thinking for Software Testers

Michael Bolton, DevelopSense

TOPICS COVERED

CONCURRENT SESSION TRACKS

customize your learning experience with
these 1-hour topical sessions.

Test Management
Test Techniques
Agile Testing
Mobile Testing

Metrics
Test Automation
**Special Topics
and More!**

COME JOIN US!

WHO SHOULD ATTEND

Software and test managers, QA managers and analysts, test practitioners and engineers, IT directors, CTOs, development managers, developers, and all managers and professionals who are interested in people, processes and technologies to test and evaluate software intensive systems

BONUS SESSION FRIDAY, OCT. 28

WomenWho | Test

Women Who Test is a full-day event on Friday, Oct. 28 for women to network with other women who work in software testing. It is also a day to learn from and be inspired by each other. The program will cover testing topics and will support women's personal and career journeys.



Alison Wade, TechWell Corp.

Women Who Test Program Chair



OCTOBER 26-27 THE EXPO

*Visit Top Industry
Providers Offering the
Latest in Testing Solutions*

Learn More: <https://well.tc/starcanadabsm>

PITFALLS OF DEVELOPING FOR THE IoT

BY LEV LESOKHIN



THINKSTOCKPHOTOS.COM

Today, if you're a technology leader and work for a public company without an Internet of Things (IoT) strategy, you can wave goodbye to your share price on its way down. IoT is no longer a nascent dream. By 2017, IDC analysts predict spending on IoT technology and services will exceed \$7.3 trillion. [1] Global brands, such as Intel, already have announced significant changes to their business to focus on IoT, and as more devices "connect" the lines of autonomous provisioning, management and monitoring will continue to blur.

Is IoT Just a Fad?

Putting the hype aside, one of the most important conversations to emerge lately relates to the tactical elements of IoT. What do organizations need to address in development to make this a successful technological shift? Without precise execution, IoT could turn into a nightmare (remember the movie *Minority Report*?) Devices are getting smarter—talking to each other and cutting out the unreliable human elements, which result in higher quality and greater productivity. Embracing and successfully managing all of the technological complexity that comes with IoT are the most important steps toward its success.

To twist the famous words stated in *Forrest Gump*, "Smarter hardware is as smarter software does." This implies that IoT products and services will only be as good as the software behind them. This isn't creating a new set of problems; software is already pervasive. Instead, the growing ubiquity of IoT just magnifies the potential impact of problematic software, and that is where the trouble begins.

Obstacles in Transitioning to IoT Development

First, many companies investing in IoT are not traditionally involved in computing. For example, today more than 50 percent of IoT activity is centered in industries such as manufacturing, government (smart city), and consumer products. [1] Some of these organizations suffer from a lack of proficiency in putting together such a dynamic software capability. This is often exacerbated by the fact that engineers, especially those with knowledge in connected device development, are in high-demand. The other issue is once an IoT competence is established, organizations then have to figure out how to manage the software. This results in a trifecta of problems. Even traditional embedded software developers haven't caught up with all the practices needed to develop Internet-connected applications. How will the rest of the current software industry manage?

Second, embedded software components have to interact safely with other Internet-facing components. Although an application can operate on secure subnets, access will be restricted to users of the same subnet. That may work for some business models, but it is not a viable solution for anyone who wants to access the global Internet. As a result,

developers have to understand how these connected components will interact in order to ensure security, reliability, and efficiency. This is a common problem in IT but new in device software development.

Third, IoT exposes developers to problems and capabilities that are, for the most part, already well known in traditional computing but how to counter them in a connected world is still relatively virgin territory. For example, enterprise and web developers are very familiar with the need for robust security against local and remote attacks. The notion of input validation as a first line of defense is well accepted in connected systems today. However, IoT development expands the scope of those concerns. Embedded, device, and mobile developers need to start considering security challenges such as input validation during development. Otherwise, it will be cost prohibitive to redesign onboard systems to include these defenses after they have shipped to customers.

What about Security and Product Quality?

One of the greatest concerns in IoT is security, and how software engineers address it will play a deeper role. Security needs to be tackled at the start of the design phase, making requirement tradeoffs as needed. During code construction, security needs to be baked into the process—making it more foundational than a mere "bolt on."

Due to the large amount of complex data being exchanged, manufacturers will have to build in privacy options that can be invoked at the expense of additional functionality. This comes back to designing in security during the requirements phase of software development.

Security is highly correlated to software robustness. It may take a little bit more time to design and build robust software upfront, but secure software is more reliable and easier to maintain in the long run. CRASH Report 2015 shows a high correlation between security and robustness as shown in table 1. [3]

The CRASH Report results suggest that one third of security problems are also robustness problems—a finding that is borne out in our field experience with customers. In my view, there's a relationship between the security budget and the overall budget for software quality.

Despite software developers' best intentions, management is always looking for shortcuts. In the IoT ecosystem, first to market is a huge competitive driver, so this could mean that quality and dependability are sacrificed for speed to release. Device manufacturers have learned the hard way that putting "glitchy" software on devices is asking for trouble. Poorly written software continues to be one of the greatest safety issues today (e.g., Jeep Cherokee's car hacking fiasco of 2015).

Sometimes glitches happen because developers are under pressure to get products or software updates released. Consequently, they end up burning the midnight oil to fix bugs that should have been discovered during the development

Pearson Correlation Coefficients	Robustness	Performance	Security	Transferability	Changeability	Lines of Code
Robustness		.31	.60	.58	.62	.15
Performance	.31		.22	.36	.37	.00
Security	.60	.22		.27	.13	-.09
Transferability	.58	.36	.27		.55	.00
Changeability	.62	.37	.13	.55		.07

CRASH Study 2015

n=1316 applications, 212 organizations, 706 million LOC

Table 1: Correlation among software characteristics (CRASH Study 2015)

process. Third-party components help offload some of the burden, but with more complexities and upkeep in IoT, components are expected to be maintained and updated to address problems at a much faster pace.

The Four Fundamental Practices in IoT Software Development

To meet demands, avoid pitfalls, and achieve success in the growing IoT marketplace, organizations need to adopt four important practices for IoT software development: review, assessment, responsibility, and advocacy.

Review: Proper code review and repetitive testing need to be a priority. Manufacturers must communicate this message to software engineering teams and call for stricter software quality measures. The high complexity of IoT applications leaves software susceptible to security lapses and software quality failure. One bad transaction between an application, a sensor, and a hardware device can cause complete system failure. Organizations just can't afford that.

For some developers, this environment is quite familiar, especially if they are running mission critical systems for a utilities provider or a bank. However, ordinary app and device software developers may find themselves engaging in the same degree of structural quality analysis and code review required to develop airline autopilot systems.

Assessment: Continuous deployment in the connected world is business as usual. Updates occur non-stop and are often pushed multiple times a day. The quality assurance burden on the software that interacts with IoT devices is greater than ever. If the software isn't continuously monitored and the code evaluated, failure is almost guaranteed.

Responsibility: Management must take responsibility for quality assurance. Any manufacturer that doesn't have a set of analytics to track its software risk—be it reliability, security, or performance—is negligent in its responsibility to customers and other stakeholders. Management needs to lead by example and communicate the direct link between software quality and security. It's in their best interest, too, since security vulnerabilities caused by poor coding or system architectural decisions can be some of the most expensive to correct.

Advocacy: In addition to measurement and analytics, a cultural shift to include education needs to occur. Developers and management collectively need to spread the word in the community about standards. Significant strides have been made in creating initiatives for manufacturers and IT departments to consistently measure the quality of their software.

In 2015, the Object Management Group (OMG) approved a set of global standards proposed by the

Consortium for IT Software Quality (CISQ) to help companies quantify and meet specific goals for software quality. [4]

CISQ was formed as a special interest group of OMG to create standards for automating measures of software size and quality attributes (e.g., security) at the source code level. While software quality standards have existed for a long time, the traditional ISO standards only specified the categories of what should be measured—not how to actually measure it.

What CISQ has done is to define in detail how these characteristics should be measured. These measures were designed for use by IT organizations, IT service providers, and software vendors in contracting, developing, testing, accepting, and deploying software applications. CISQ's measurement standards include security, reliability, performance, and maintainability. This allows businesses to certify the quality of codebases and IoT networks.

Succeeding with IoT

Sexy consumer applications (predictive coffee makers, self-driving cars) and cyber attacks (DDOS assaults launching from refrigerators) still dominate the IoT headlines, but the deeper business value is starting to emerge. The McKinsey Global Institute report points out that mature IoT systems will take the guesswork out of product development by gathering data about how products, including capital goods, actually function and how they are used, rather than relying on customer focus groups. [5]

That is certainly a game changer requiring development teams to reduce risks in the software they engineer now. Understanding the importance of a secure architecture foundation and insisting that developers comply with industry standards will be the first line of defense.

After that, the rest is up to you. **{end}**

I.lesokhin@castsoftware.com

**Sticky
Notes**

[Click here](#) to read more at StickyMinds.com.

■ References

NEWSLETTERS FOR EVERY NEED!

Want the latest and greatest content delivered to your inbox every week? We have a newsletter for you!

- **AgileConnection To Go** covers all things agile.
- **CMCrossroads To Go** is a weekly look at featured configuration management content.
- **DevOps To Go** delivers new and relevant DevOps content from CMCrossroads.
- **StickyMinds To Go** sends you a weekly listing of all the new testing articles added to StickyMinds.
- And, last but not least, **TechWell Insights** features the latest stories from conference speakers, SQE Training partners, and other industry voices.

Visit StickyMinds.com, AgileConnection.com, CMCrossroads.com, or TechWell.com to sign up for our weekly newsletters.

SOFTWARE TESTING TRAINING WEEKS

Featuring up to 16 specialized classes, SQE Training's Testing Training Weeks offer software professionals the opportunity to maximize their skills training by building a customized week of instruction. Courses are offered for QA, test, and other software professionals across the development life cycle and include topics such as agile, DevOps, mobile, test automation, test management, and more.




UPCOMING DATES AND LOCATIONS

August 22–26	Dallas, TX	October 17–21	Tampa, FL
September 19–23	Washington, DC	November 7–11	San Francisco, CA



2016 Testing Training Week Schedule

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
Software Tester Certification—Foundation Level			Mastering Test Design	
Security Testing for Test Professionals*		Integrating Test with a DevOps Approach		DevOps Test Integration Workshop
Fundamentals of Agile Certification—ICAgile		Agile Tester Certification		Agile Test Automation—ICAgile
 Green background indicates courses pre-approved for Project Management Institute PDUs.		Mobile Application Testing*		Mobile Test Automation Workshop*
Essential Test Management and Planning*		Measurement & Metrics for Test Managers*	Leadership for Test Managers*	Test Improvement for Agile*
Risk-Driven Software Testing*		Performance, Load, and Stress Testing*		*Not available in Dallas.

SMARTBEAR

We help build quality applications for the
connected world



CODE

COLLABORATION

Peer code and documentation review



TESTING

Functional testing, performance testing
and test management



PERFORMANCE

MONITORING

Synthetic monitoring for API, web, mobile,
SaaS, and Infrastructure



API

READINESS

Functional API testing, load testing,
API virtualization & security testing

RECOGNIZING #WOMENINAGILE

How Men and Women Perceive Female Involvement in the Agile Community

by Natalie Warnert

It's a great time for girls to be involved in technology. Clubs like Lego Robotics and Girls Who Code are prevalent. Toys and store aisles are less gender-specific and try to attract all children to science, technology, engineering, and math (STEM). Although there is no reason girls shouldn't be more involved, interested, and advancing in technology fields, it's still not the reality.

True, progress has been made from the previous generations of women who entered into technical fields by taking male-dominated classes. But women are still fighting the battle to gain a voice in a largely male field. If the women who are currently working in technology do not define and increase their presence, the girls of today will experience the same problem: fewer technical female role models willing to sponsor and mentor them.

As a woman in a STEM profession, I pondered the reasons we often show up without showing up, watch from the back without participating in the front, and listen instead of lead.

Setting Up My Research

The lack of women's involvement is apparent in the agile and technology communities, which have been at the center of my career journey. With women still a minority in information systems classes and corporate employment, the lack of diversity is glaringly obvious.

I have observed lower female involvement patterns in agile community activities, including submitting presentations for conferences, authoring subject matter articles, attending and participating in user groups, earning certifications, and blogging on agile-related topics. This lack of involvement formed the foundation of the thesis research for my master of arts degree in organizational leadership. My thesis attempts to answer

two questions: "What are the reasons that prevent female agile practitioners from achieving higher levels of involvement in the agile community?" and "What strategies can overcome them?" I conducted qualitative interviews with women with various levels of involvement, as well as quantitative surveys through social media that were open to agile community members of both genders.

Research Results

The initial results of my research show 10 percent of men and 19 percent of women believe gender involvement is equal and is not an issue. However, of the population of survey respondents, only 40 percent were men and 60 percent were women. On a 0 to 10 scale, men self-report higher levels of involvement than women, with a 95 percent confidence interval (see table 1).

	Women's involvement level	Men's involvement level
Mean involvement (0-10)	4.47	5.67
Observations	57	36
P (T<=t) one-tail	0.0317	

Table 1: Mean difference in men's and women's involvement levels on a 0 to 10 scale

Reasons for Lower Involvement

Participants were asked "What prevents you from being more involved in the agile community?" and were instructed to choose only the two most significant reasons. The survey results are shown in figure 1.

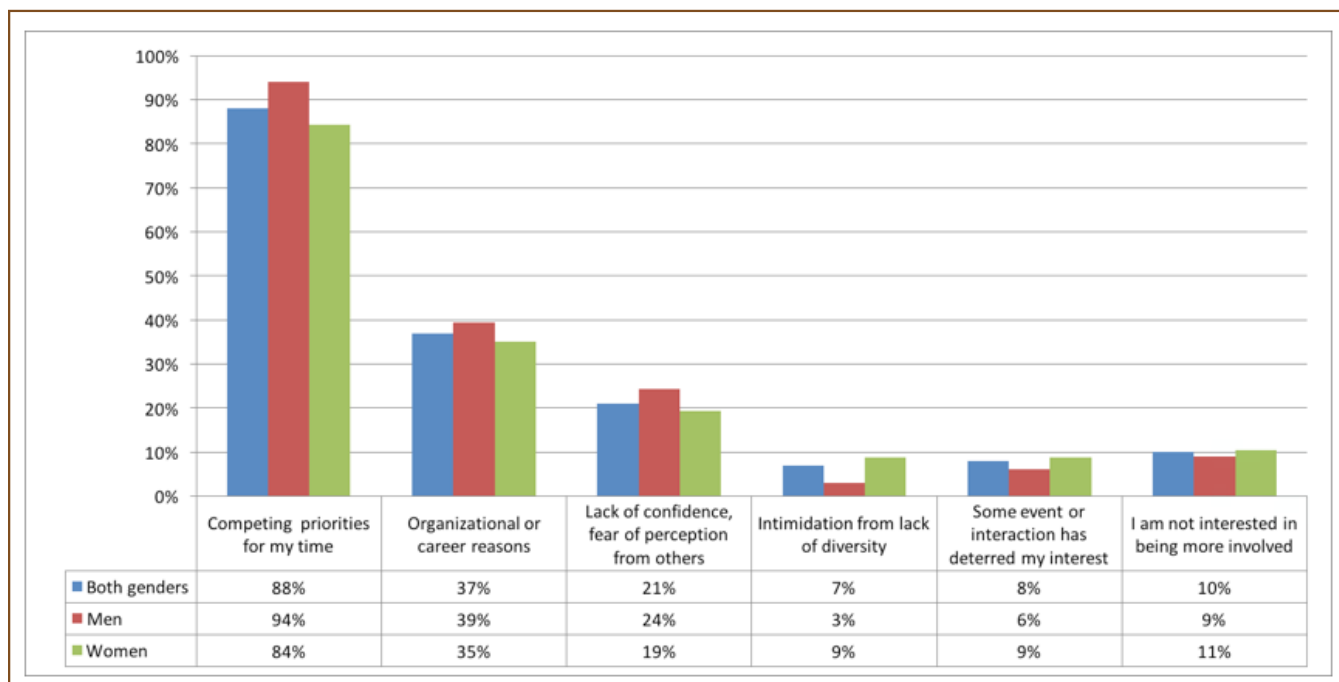


Figure 1: Self-reported reasons that prevent women and men from higher levels of involvement in the agile community

The overwhelming majority of both genders state that competing priorities for their time prevent higher involvement levels. This is not a surprise, though men report this as one of their top two reasons more frequently than women (94 percent and 84 percent, respectively).

The second most given reason for lower involvement levels from both genders is organizational and career constraints—mainly, a lack of company encouragement and support for involvement activities. The survey results show that 60 percent of companies provide financial support for employees to be involved in the agile community (e.g., paying for conferences, certifications, training, and user groups; allowing employees to use work time to attend the activities; etc.). When comparing the mean self-reported involvement levels from both genders, the involvement levels are statistically significantly higher for those individuals whose companies provide financial support (see table 2).

While women are often thought of as being less confident and men are perceived as being overconfident, the survey shows different results. Men cite a lack of confidence more frequently (24 percent) than women (19 percent). It also seems that lack of interest is not an important factor to lower involvement levels, with only 10 percent of both genders citing this as one of their top two involvement hindrances.

	Involvement with financial support	Involvement without financial support
Mean involvement (0–10)	5.31	4.12
Observations	52	33
P (T <=t) one-tail	0.0378	

Table 2: Agile community involvement level correlated with company support of men and women combined on a 0 to 10 scale

Perceptions of Women Involved in Agile

The next question asked participants, “When you think about the agile community, in your opinion, why are men more involved than women?” Participants were instructed to choose only the top two reasons. The results in figure 2 show apparent discrepancies between the self-reported results of both genders and the perceptions when thinking about the female agile professional population.

The results show staggering differences in self-reported reasons and gender perception-based reasons. Individual women overwhelmingly see themselves as superior when examining their reasons for lower involvement compared to looking at women as a population. It is almost as if women are saying,

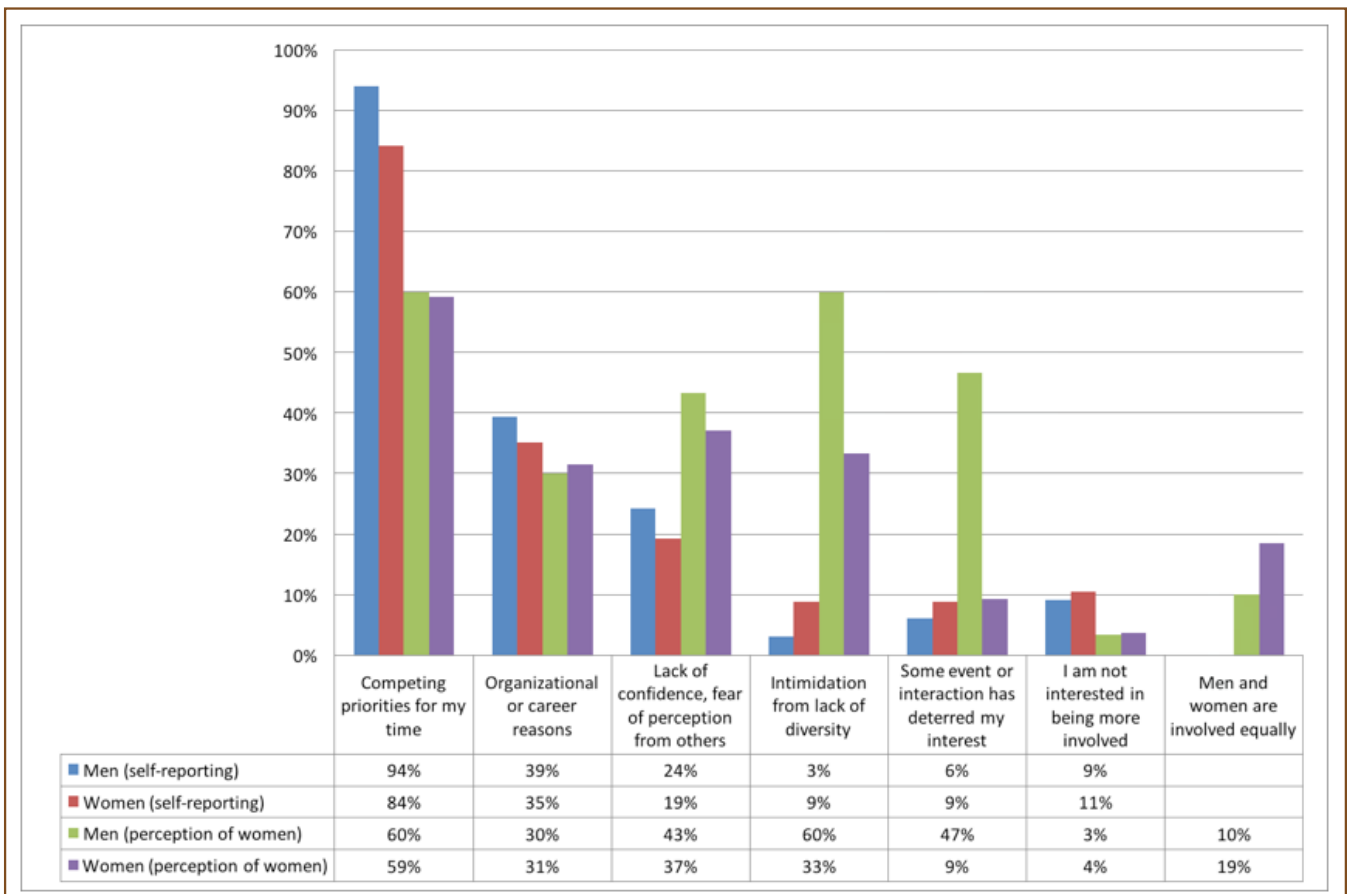


Figure 2: Perceived reasons that prevent women from achieving higher levels of involvement in the agile community

“That’s an issue other women have, but not me.” A higher percentage of women also think men and women are equally involved in the agile community, which was disputed earlier in the article (see table 1).

Men’s answers reflect the majority gender’s perception of the minority gender. When asked specifically about women, men mirror women’s responses, but at an even higher percentage. In fact, the study shows men perceive women to be intimidated by a lack of diversity (likely gender diversity) at the same percentage (60 percent) as both genders’ most popular reason: competing priorities for time. Additionally, men perceive women to have diversity-based intimidation as an involvement deterrent at almost double the percentage that women do (60 percent compared to 33 percent). The next most popular answer, which 47 percent of men chose in their top two reasons for women’s lower involvement levels, relates to an event or interaction that deterred women’s involvement interest.

The Final Analysis

This research reveals major issues with how individuals perceive themselves, their gender, and the opposite gender. Women and men perceive themselves as superior to the female population as a whole by reporting that the more negative and stereotypical lower involvement reasons do not personally affect them to the same extent as they do women as a collective group.

Why do women think of themselves as empowered and their collective gender as underpowered? Why are women competing against each other for the one slice of the pie they currently have, when there is an entire other part of a pie currently occupied by men that can be shared? Other women are not the enemy, and neither are men. The real enemies are the embedded perceptions we harbor as a society. The enemy is how these implicit gender biases change our behaviors toward ourselves and others.

Why do both men and women think women suffer from a lack of confidence? Why do men assume women are intimidated by the lack of gender diversity? Why do men think women’s involvement has been deterred by an interaction within the agile and technology communities? How do these assumptions influence men’s and women’s communication and professional relationships? The assumptions, biases, and resulting behaviors—deeply rooted in societal perceptions and their associated interactions—need to be challenged.

Call to Action

Changing society’s perceptions will take immense time and overwhelming effort. We must ask what we can do now to make a small difference that can expand to effect change. We need to return to the personal, human aspect of involvement and communication.

I have found that a few simple steps taken by individuals can make a huge difference. Extend personal, direct, spoken invitations for others to participate in agile community activities, such as meetups. Become more involved yourself and encourage companies to support you and others in these en-

deavors. Invest the time in building community relationships to collaborate together. Expand your children’s horizons to include STEM educational activities.

This is not only a women’s effort. If men and women ally on this quest, the results will benefit everyone. The more both genders are involved, the more quickly the effects will be recognized and the greater the overall impact will be. Efforts like #HeForShe provide ideas to help women and men become partners instead of adversaries.

When more women and men are involved in these strategies, the effect will ripple throughout the agile and technology communities. This should result in a positive change in the audience, diversity, and quantity and quality of ideas.

Conclusion

Getting women more involved in agile is everyone’s issue. Women cannot solve the problem themselves. When levels of women’s involvement increase, ideas and knowledge can flow, benefitting everyone—the team, the company, and, ultimately, the customer. **{end}**

info@nataliewarnert.com



Development Testing

Prevent Defects Before They Become Features

Static Analysis, Unit Testing, Coverage Analysis,
Peer Code Review, Runtime Error Detection

Continuous Testing

Test Features Before they Become Defects

Service Virtualization, API Testing, Load Testing
Test Data Management, Performance Testing

www.parasoft.com/bettersoftware2016



**LIVE
VIRTUAL
TRAINING**

**ATTEND LIVE,
INSTRUCTOR-LED
CLASSES VIA
YOUR COMPUTER.**

Live Virtual Courses:

- » Agile Tester Certification
- » Fundamentals of Agile Certification—ICAgile
- » Testing Under Pressure
- » Performance, Load, and Stress Testing
- » Get Requirements Right the First Time
- » Essential Test Management and Planning
- » Finding Ambiguities in Requirements
- » Mastering Test Automation
- » Agile Test Automation—ICAgile
- » Generating Great Testing Ideas
- » Configuration Management Best Practices
- » Mobile Application Testing
- » and More

Convenient, Cost Effective Training by Industry Experts

Live Virtual Package Includes:

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

SQETRAINING.COM/LIVE-VIRTUAL





MAKING THE MOVE TO

Product-Driven Process

by David Hussman

SHANE STOCK PHOTO.COM

Fifteen years ago, I was weary of programming and looking for a change. I was working in a company that, even though it built products, was run more like an IT shop. Our discussions were more about “Will that project finish on time?” and less about “What are our customers doing?” Being project-focused, our team was often building for the future—attempting to create the framework to save all projects.

Although my job title was something completely different, I was essentially the technical lead. I had a small team working hard and struggling to stay on schedule. In reflection, our challenges were classic: failure to break down huge use cases, over-designing for the future, trying to incorporate all the newest technology, always waiting to integrate, and doing little to no real testing. The diminishing morale of the team combined with the pressures on the project told me it was time for a change.

Connecting the Dots

In hindsight, our failure to build a meaningful, great product was at the heart of my frustration. My prior job was with a small software technology company where I wore many hats: I was responsible for building a product that I also demoed to customers and prospects at trade shows. With the combined role of programmer, presenter, and sales engineer, connecting with customers was a way of life and the lifeblood of the product. Learning to tell stories that added context was a natural way to make connections and learn about customer needs.

All that I loved from my past job was missing at my new job as a technical lead. We were working off use cases that were filled with details but lacking in context. We were so enamored with what our new technology could do that we lost sight of what we should do for our customers.

In my search for new ideas for my team, I stumbled on Extreme Programming (XP), one of the many lightweight methodologies that would become part of the genesis of the agile movement. I read a bit about XP and quickly put the ideas into action. We went from a group of people working next to each other to a test-infected, highly collaborative team. Our behavior and motivation changed to discussing and addressing issues, automating integration, automating testing, and trying to adopt a customer-driven approach, even if our customer was an internal business analyst.

Leading and Coaching Agile Adoptions

After a short time using XP, my team was happier, more motivated, and producing results. Other teams and other leaders came to see our collaborative workspace and information radiators. At the same time, our CTO decided more teams should try XP. A small collection of experienced practitioners was rebranded as the incubator and given the mission to help other teams adopt agile methods by working directly with them for a period of time. We did not know we were leading what is today known as an agile transformation.

Filled with the passion of success using agile methods, we were off to share with others what we had experienced. After a handful of successes, we started to stumble as we simplistically tried to recreate our experience in dissimilar situations.

Although able to introduce agile practices, we did not always reproduce previous success. It was clear to us that the context was, as it always had been, an essential ingredient in a successful transition to agile methods. Teams were inspired when they could see the impact of their efforts. For these teams, agility grew into a learning tool instead of a process to follow.

This over That: Rebels with a Cause

Around the same time, and as part of my introduction to the agile world, I was fortunate to have a white paper accepted to present at an early XP conference. I was inspired by the confidence, experience, and openness of the critical thinkers I met, many of whom became signatories of the Agile Manifesto.

The *this over that* format they chose for the manifesto was simple and powerful, reshaping the way we thought and worked.

I used the *this over that* comparison with teams I coached. Working directly off the pages of the XP playbook, being a coach meant sitting with teams, challenging them to practice continuous integration, adopting a test-driven focus, and employing the user stories during what was appropriately called *the planning game*.

Coaching teams toward XP meant asking them to change the way they collaborated. Of the many significant and often unnamed changes, we valued learning from delivery over failing to get things done. We talked about starting by finishing, and we were merciless about integrating early and often while being mindful that all the automated tests needed to run successfully.

We did not know that practices like *nuke and pave* (deleting everything before each new build) would eventually become a key component of today’s DevOps movement. As we worked to create deterministic environments where all our tests were passing, we started to learn that our code did not work downstream in the production environment.

A few of us were brave and humble enough to leave the safety of our collaborative spaces and head out into the land of integration testing and operations.

Emergent Behaviors and the Importance of Product

As the agile movement grew, more people were using agile methods in more varied context. The language of *project* was replaced with the language of *progress*. Questions like “Did the build pass?” and “What did the customer say about that story?” abounded.

Agile transformations were in progress and teams were getting more work done, more often, with fewer issues and better quality code.

For many of us, the confidence of knowing that work could be completed gave us the courage to question if what we were doing was meaningful. I once again sought out actual end-users to interview, trying to understand their experiences as part of building the right thing. I again felt like that programmer who stood on the trade show floor, listening to experiences and ideas from customers. I started to explore more ideas around product validation over iterative progress, and this led me to start practices that I call *product transitions*.

Transitioning to Products

Product transitions are more and more the focus of my coaching. Helping companies transition to a product-driven mindset includes language changes, perspective changes, build and deploy changes, and organizational changes. As an example, I now prefer the term product coaching rather than agile coaching. I no longer care whether teams practice Scrum, XP, kanban, or the next agile brand that emerges. I have always been enamored with the impact that came from helping people build something they could be proud of.

Helping teams see the effect of their work as it changed people's lives is the bonding agent that lasts.

I know many process evangelists who are bonded to their process over their product. While this might be a comfortable place to start, I find it to not be sticky across time. When people are bonded around process and then grow weary of the process, they often move on to a new process without addressing root issues or chasing valuable opportunities, or they backslide into what is comfortable, even though it may be dysfunctional or not meaningful.

Conversely, teams who are connected to the impact of what they produce tend to be stronger and more resilient to adversity. Nassim Taleb, the author of *The Black Swan*, might call them the antifragile teams (also the name of his latest book). I now call them product teams, many of which are using agile practices and methods like Scrum or kanban.

Product-Centered Principles

To help teams and companies with a transition to a product focus, I like to introduce a set of product-centered principles. Riffing off the manifesto, I present them in the simple *this over that* comparative format. Let's use this group as a starter set:

- Learning over delivery
- Experiments over iteration
- User experiences over user stories
- Continuous learning over continuous integration

While these are only a few of the comparisons I use, they are a meaningful selection for you to think about as a starting place. (Please keep in mind that *this* does not replace *that*. *This* is more valuable than *that*, and *this* often depends on *that*.)

Learning over Delivery

If you've spent time in the trenches working with teams—especially collections of teams working on the same product—things are often left undone or take longer than expected. If you are gathering meaningful evidence, you also know that there are many reasons that work is not completed as planned—with many reasons having little to do with the people actually doing the work.

Forces outside teams often impact delivery and learning from delivery. I've always looked to iteration boundaries as windows for learning first—with quantity delivered running a distant second. I'm always sad to see people, especially people calling themselves coaches, hammer on people with damaging language like “You missed your sprint commitments.” When I meet people who've been hit with the dogmatic Scrum hammer, I try to introduce a culture of *learning over delivery*.

Helping teams *learn to learn* allows them to change their perspective from delivery to learning, which is one step in opening the door to learning in product development over merely getting things done. For the experienced and more confident teams, it is possible to challenge them to explore what they might be able to learn without always building to production.

For example, it might help asking, “What can we learn outside the code?”

Experiments over Iteration

Building on learning over delivery, helping people be more open by framing iterations as experimentation is another step toward transitioning to products over process. If the word *product* is uncomfortable at this point, feel free to think of product as the things you produce that impact others.

Asking teams to talk about measuring the impact of completing stories is a simple and concrete way to subtly introduce experiments. If your team uses a kanban board or story wall, you can seed experiments by adding a column called Validated to the right of the Done column.

Going further, asking teams to discuss how stories will be validated before pulling one into the in-progress column is similar to framing a hypothesis for an experiment before starting the experiment. I find that this often results in doing less of what is unknown and engages more people in thinking about value over delivery.

User Experiences over User Stories

Once upon a time, stories were called *story cards* and were meant to be conversation starters, not merely shorter requirements documents. As people struggled with maintaining a customer-based discussion, the word *user* was appended to *story*, and user stories were born. In some cases, this was documentation over collaboration taking root like a weed growing in a newly tilled agile garden. It was a sad irony to be certain.

Where users sometimes promoted a customer focus, it did not solve—and sometimes exacerbated—the fact that many user experiences cut across one or more stories. Similarly, using the term *epic* often drives teams toward emphasizing smaller stories over user experiences.

Arranging stories in a left to right style, now known as a *story map*, provided a visualization of interactions from which user experiences could emerge. As a product coach, I use this visualization to help teams openly discuss, “Where do you want to take your customers?” This quickly challenges the team to move past delivery of working code (passive) to active validation of customer value produced (active).

Continuous Learning over Continuous Integration

It is exciting to see teams and companies growing their continuous integration and continuous delivery skills, but it's even more exciting when they are doing it to accelerate product learning and customer development. For many months, we have been part of a large-scale DevOps *dojo* where teams learn continuous delivery tools and skills as a way to promote continuous learning.

For companies new to continuous learning, we tend to introduce concrete skills around continuous integration as a few of the many tools needed to move to a continuous delivery model that feeds continuous learning. If you can't integrate, you can't deploy; if you can't deploy, you can't test; and if you can't test, you can't learn.

If you view environments and deployments as something you can control and automate, invest in continuous integration and continuous delivery as tools for helping teams learn more quickly whether they are building the right thing.

Change Is Still Challenging

So, who is succeeding on the journey to products over process? In general, product companies have an advantage over IT shops in that the idea of product is part of their vocabulary and focus.

That said, many product companies are still overly focused on agile methods as tools for getting more done. They have not yet adopted the perspective that no product idea is valuable until proven to be valuable. Both value and viability are best measured with customers.

The move to a product-driven culture is easier when the product-to-team mapping is one-to-one. For larger efforts, where many teams are building one product, the challenge to move to a product-driven mindset is constrained by the ability to integrate across teams, as well as the mindfulness of the way work is decomposed and restructured so that ideas can be validated.

Teams and companies where delivery is more constant are more likely to make the move toward product learning fed by delivery confidence. When teams start working in

smaller units of work, the size of the failures decreases to where they might not view them as failures and instead simply call them learnings or experiments.

I often teach teams that the difference between failure and learning is the cost, either in money or ego. If you're ready to spend less time failing and make a greater investment in learning, then you're ready to move toward products. You may even be ready to start a product transition for your team or organization. **{end}**

david.hussman@devjam.com

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. The following is a sample of some of the great content you'll find. Visit TechWell.com for the full stories and more!

Test Automation—A Non-Expert's Perspective

By Dale Perry

I have been involved in IT for four decades now. Starting as a systems programmer in the '70s, through database management, network management, project management, and now I specialize in testing. During the past five years, I have noticed a strong trend in many organizations to push toward automating test execution.

I am not a tools expert, and I do see value in the application of automation to testing. However, I notice that many organizations fail to get real value from their automation efforts. Looking at automating test execution, I see two distinct areas of focus, testing at the technical level (component and technical integration) and testing at the functional level (system or acceptance). Each level has different requirements and focus.

<https://well.tc/38qt>

Pick a Chicken: How to Prioritize and Get More Done

By Payson Hall

When I was eight, my family moved from the suburbs to a small farm. At some point my mother sent me out for the first time to catch a chicken for dinner. There were perhaps twenty chickens in a fenced pen with no place for them to go, but I made the mistake every rookie predator makes when hunting chickens: I ran in among the birds, trying to grab whichever one was closest.

Chaotic hilarity ensued. Chickens can't fly very well, but if you run at them, they scatter in a random pattern. I made several attempts, coming up empty-handed each time.

Finally Mom or Dad took pity on me and said, "Pick a chicken. If you don't pick a particular chicken, they will all get away. Pick one and chase it till you catch it."

<https://well.tc/38qb>

The Test Expert's Role in DevOps

By Michael Sowers

It wasn't long ago that those of us who dedicated our professional careers to helping our organizations deliver better software were asking how we fit into this new world of agile development. While individual companies and the software engineering community continue to adapt and decide whether agile is right for their businesses, there's already another important advancement in our world: DevOps. Some see this as simply the next evolution of agile.

<https://well.tc/38ck>

Learning On: Making Time for New Software Skills

By Steve Berczuk

Software development is a fast-paced profession. Technologies you work with and the skills you need are constantly changing. Ongoing learning will help you remain relevant as the industry evolves, as well as be more productive at your job. However, it can be hard to find time for education.

It can be tempting to try establishing expertise in new areas quickly. If you are not careful, you may reach a plateau where you think you are an expert. Erik Dietrich describes how developers end up as "expert beginners" rather than moving on to being true experts.

His claim is that, while the normal progression is from novice to advanced beginner to competent to proficient and, finally, to expert, some people take a detour after they become competent to what he calls expert beginner.

<https://well.tc/38q5>

Fine-Tuning and Expanding Your Mobile Test Plan

By Melissa Tondi

When mobile debuted, it became one of the bigger disruptors in the technology world. Many companies struggled to quickly go to market with mobile product offerings—and ended up sacrificing some engineering practices throughout the journey. As most of us have experienced, when new technology is introduced, test teams generally have a period of scrambling to figure out how to support those engineering efforts, without necessarily having the luxury of an increased budget for tools and additional staff.

<https://well.tc/38xs>

Scaling Product Agility: More Product, Not More Process

By David Hussman

I was working with a group building a fast-growing product, and we went from having one team to a large collection of teams in no time. Each team told us its needs were the most important. As the person pushing agile methods, I had to improvise and create the initial practices for what I now call scaling product agility.

My focus was on facilitating workflow for all the teams and producing more of what they really needed (and less of what they didn't). I decided to organize a cross-team product discovery cadence to discuss their mutual and specific needs.

<https://well.tc/385r>

IoT and the Wisdom of Mobile

By Steven Winter

Industrial revolutions come and go. Their impact is measured over time by the progress their disruptions bring, and typically, they are measured in how we reacted and evolved due to that disruption. Personal computing, the Internet, and mobile devices are all prime examples of such massively disruptive forces, and now the Internet of Things (IoT) has taken the world by storm with its tentacles of connectivity growing exponentially by the minute.

Sound a bit dramatic? Well, it's not. With over 12 billion—yes, billion—devices connected today and 5,000 more wired up by the time you're done reading this article, the hyper-connectivity of IoT is here with its promise of dramatically increasing the quality of our lives.

Rough estimates give the IoT impact an economic impact more than \$11 trillion per year by the year 2025. This is relatively the same amount of fiscal value mobile will have by then, but mobile's been around the block a heck of a lot longer. To say IoT is the greater disruptor is an understatement.

<https://well.tc/38gn>

Do You Design Your Software Process for Flexibility or Repeatability?

By Johanna Rothman

Back in the '90s, everyone was into repeatable processes for software.

What that often meant was we were supposed to do all projects in exactly the same way, regardless of the type of project. We were supposed to document what we did and deliver the same documentation each project. It didn't matter how much innovation we needed, how many experiments we needed, or if we needed to iterate on the architecture for one project and not another. We were supposed to treat each project the same way.

Repeatability was supposed to provide us with a guaranteed outcome every time—if we could just define the right process. (Does this sound like requirements to you?)

I see the same problem with agile now.

<https://well.tc/3b5z>

How Continuous Integration Can Save Your Team Substantial Time and Effort

By Josiah Renaudin

Everything in life has a feedback loop, if you ask principal consultant and member of the core team at Agile Artisans, Jared Richardson. A leads to B. B leads to C. What you put in leads to what you produce, and while that's something you can apply to just about anything outside of work, it's especially critical when it comes to software development.

Continuous integration is all about the feedback loops between your developers, testers, product owners, customers, and everyone involved in your organization. That's great to write as if it's gospel, but what can continuous integration and continuous testing do for you right now to improve efficiency?

<https://well.tc/3ang>

What to Do When Bugs Are Found—Based on When They Are Found

By Hans Buwalda

Action Based Testing (ABT) is based on the importance of test design to drive automation success. It uses a modular keyword-driven approach, which means that tests are organized in test modules and built of sequences of actions—each consisting of an action name (keyword) and zero or more arguments. In our TestArchitect tool we define these in a spreadsheet-like format that is easy to work with. Test modules can contain multiple test cases that need to fit into the scope of that particular module. The test cases can form a narrative in which each test case can set up the preconditions for the next one.

The development and automation of test modules fits well in sprints. Typically a sprint will start with higher-level test modules that are at a similar level as the user stories and acceptance criteria. Once the sprint starts creating the detailed UI, the lower level interaction test modules can be created as well.

<https://well.tc/3b59>

IoT Security Concerns for Quality Assurance Teams

By Sanjay Zalavadia

The Internet of Things has been touted as the latest potential trend, but the fact is that the IoT is already here and is impacting businesses and consumers everywhere. Any object that is Internet-enabled and has machine-to-machine communication capabilities is part of the IoT—including smartphones, cars, and fridges. In addition to the typical Internet-related security concerns, applications are being made specifically for these devices, all of which bring about additional security concerns that quality assurance teams need to consider.

Since the IoT is still relatively new, organizations have been slow to set standards. However, as more consumers look to become connected, businesses must ensure that their IoT products and services are secure.

According to a recent Kelley Blue Book survey, not only are consumers unaware of and unconcerned about the security threat to connected cars, but a majority believe that manufacturers need to offer software to protect their vehicles. Unfortunately, the report also showed that companies are failing to implement common security measures into their IoT devices and services, making them viable targets for hackers. QA teams must ensure that the software lives up to industry expectations and that ample protections are in place to prevent unique IoT vulnerabilities.

<https://well.tc/3b5y>

Cracking the Code on Millennials

It is time to dispel the perception that the latest generation of developers aren't innovators. Millennials are just what we need.

by Jason Garber | jason@promptworks.com

We all know the stereotypes: Bearded failure-to-launch with a man bun. Unmotivated, self-absorbed slacker. Parents' basement-dwelling college graduate. Part-time barista at a hipster coffee house (but just until her alternative-punk-folk band gets signed by a cool indie record label).

As Mark Twain once observed, all generalizations are false—as is this one. A few individuals may match the description above, but they don't typify the generation. The truth is that Millennials are as bright as and even more educated than previous generations. They are just as motivated, although what motivates them may be quite different from what motivates older professionals.

This is certainly true in the software community. The young coders we encounter are smart, skilled, aspirational, and hungry for opportunity. But if you want to attract them to your business, you had better understand how to reach them and what's important to them.

Recruiting: You can forget recruiters and recruitment emails. Millennials ignore them. They much prefer to make contact through networking and personal referrals from friends, colleagues, and classmates who know you or your business. With the rise of online social media groups and in-city meetups, there are plenty of creative, unconventional ways to attract young talent.

To start, you need to tap into their network. For example, many young coders are involved in free and open source software activities. Locally, we have organizations like the Philadelphia Python and Ruby user groups, new technology meetups, and open source mentorships.

Hiring: How do you convince Millennials to come work for you? Well, obviously, seeing your company's environment is important: the work areas, the break room, the bike racks, and so on. But seeing alone is not enough. Many Millennials want to experience a company—to get a sense of its culture—before committing to an offer.

One way to do this is to invite potential new employees to speak with other recent hires of a similar age with similar

skills and experience. They can hear firsthand what it's like to work at your company. We often invite prospective hires to our weekly company lunch so they can observe our company's culture themselves.

Test driving: Another experience you can offer young coders is a pair-programming session. You and the prospect sit down at one computer and complete a programming exercise. Unlike giving them a solo programming test, this allows you to see how they think as they work and lets them experience what it's like to work with you.

One of the many benefits of pair programming is the back-and-forth you engage in. Millennials want to know your expectations and be able to ask questions. If they get stuck on an issue, you can provide guidance and gain insight into how they think about the problem.

In a solo programming test, the prospect could get stymied by a small issue and not be able to complete the task successfully. Ultimately, you'd know less about them and might reject a perfectly good candidate.

For technical jobs other than programming, you can perform a similar collaboration effort by thinking through a real-life problem on a whiteboard. Regardless of the approach, getting a feel for how well the applicant thinks, collaborates, and initiates communication can be a win-win for both of you.

Managing: Congratulations, Millennials are now part of your team. So, how do you keep ambitious young professionals happy to stay? The answer lies in understanding what motivates Millennials and then providing it.

First, they have skills and want to apply them. But they also want to broaden those skills and learn new things. Create an education and skills development plan for new hires—classroom certifications, participation in conferences, and stretch opportunities on the job.

Second, they want to know that as they acquire skills and perform well, there is a career progression path that will lead to greater opportunities, responsibilities, and rewards. Setting

“But if you want to attract them to your business, you had better understand how to reach them and what's important to them.”

up individual career development plans and reviewing progress regularly are essential.

Third, Millennials want to be engaged and connected with their colleagues. Make sure you apply a personal touch and include them as members of the community. Remember their birthdays and celebrate their accomplishments. Involve them in developing fun activities they can share with you and their coworkers, whether in a morning yoga class or on an outing to a ball game.

Finally, young software professionals want to know that their work matters and they're not just writing lines of code for the sake of writing code. They want to know that what they are creating is solving real problems for your clients and making the world a better place, so be sure to share more than just the daily list of job chores to be executed. Map out a project plan that extends over several months, review it regularly, and tie those daily tasks into the broader vision of the value you provide to your clients. Because so many software businesses put the focus on the customer, include Millennials in discussions with your clients so that these new employees understand things from the customer's perspective.

Cracking the Millennial code: Getting to the heart of the newest generation of workers—who they are, what they want, and what matters most to them—shouldn't really be that difficult. If anything, Millennials are probably overly transparent about their passions, interests, and what drives them. Understanding this generation is a first and fundamental step, but it is only the beginning. To attract and keep young talent for your business, you must engage them, draw them in, and provide a rich experience that addresses their wants and desires. {end}

WANTED! A FEW GREAT WRITERS!

I am looking for authors interested in getting their thoughts published in *Better Software*, a leading online magazine focused in the software development/IT industry. If you are interested in writing articles on one of the following topics, please contact me directly:

- Testing
- Agile methodology
- Project and people management
- DevOps
- Configuration management

I'm looking forward to hearing from you!

Ken Whitaker

Editor, *Better Software* magazine

kwhitaker@techwell.com

index to advertisers

Agile Dev, Better Software & DevOps East	https://bsceast.techwell.com	Inside front cover
Infostretch	http://www.infostretch.com	11
Parasoft	https://alm.parasoft.com/bettersoftware2016	28
Ranorex	https://www.ranorex.com	2
SmartBear	https://smartbear.com	23
SQE Training - Live Virtual	https://www.sqetraining.com/live-virtual	29
SQE Training - Test Weeks	https://www.sqetraining.com/trainingweek	22
STARCANADA	https://starcanada.techwell.com	14
STARWEST	https://starwest.techwell.com	9
TCS	https://goo.gl/GmUBcZ	Back Cover

Display Advertising
advertisingsales@techwell.com

All Other Inquiries
info@bettersoftware.com

Better Software (ISSN: 1553-1929) is published four times per year: January, April, July, and October. Print copies can be purchased from MagCloud (<http://www.magcloud.com/user/bettersoftware>). Entire contents © 2016 by TechWell Corporation 350 Corporate Way, Suite 400, unless otherwise noted on specific articles. The opinions expressed within the articles and contents herein do not necessarily express those of the publisher (TechWell Corporation). All rights reserved. No material in this publication may be reproduced in any form without permission. Reprints of individual articles available. Call 904.278.0524 for details.



Digitally transforming your business? To get it 'first time right', there is a certain way.

In a fast-evolving marketplace which demands leadership that brings results, there exists a way of certainty: Tata Consultancy Services (TCS). With TCS as your strategic advisor and partner, the ever-changing new landscapes of business become new vistas of opportunity for transforming your organization.

Within this dynamic environment, does your QA and Testing function ensure you achieve your digital transformation goals the first time, every time? With TCS' Assurance Services Unit (ASU) delivering proven world-class enterprise testing experience and expertise, you can be certain of the quality and speed to market of your transformation initiatives.

Visit tcs.com/assurance and you're certain to learn more.
Or write to us at: global.assurance@tcs.com.

IT Services
Business Solutions
Consulting



TATA CONSULTANCY SERVICES

Experience certainty.

TATATATA

#ThinkAssurance Today!

