Spring 2015 www.TechWell.com

BETTER SOFTWARE™

A TECHWELL PUBLICATION

AGILE NOT WORKING OUT?
Stop making lists, start making products

SCALING DEVOPS
Continuous integration requires an entirely new approach

Adopting ALM Will Enhance the Value of Your Test Team

LEAN MEAN SOFTWARE DEVELOPMENT MACHINE

AGILE DEVELOPMENT CONFERENCE WEST

DEVOPS CONFERENCE WEST

BETTER SOFTWARE CONFERENCE WEST

THREE CONFERENCES IN ONE LOCATION

JUNE 7–12, 2015 | LAS VEGAS, NV

CAESARS PALACE | #BSCADC

Register by April 10 and save up to $400 off your registration

ADC-BSC-WEST.TECHWELL.COM

CONFERENCE SCHEDULE

Choose from a full week of learning, networking, and more

SUNDAY
Multi-day Pre-Conference Training Classes begin

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

WEDNESDAY–THURSDAY
Keynotes, Concurrent Sessions, Expo, Networking Events, Receptions, and More

FRIDAY
Agile Leadership Summit

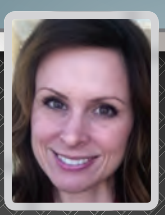
KEYNOTES

SOFTWARE DEVELOPMENT EXPERTS SHARE INSIGHT

Why DevOps Changes Everything
Jeff Payne, Coveros, Inc.

Better Thinking for Better Software: Thinking Critically about Software Development
Laurent Bossavit, Institut Agile

Lean UX: Turn User Experience Design Inside Out
Jeff Patton, Jeff Patton & Associates



Back to the (Agile) Future: Doing Agile or Being Agile?
Stacia Viscardi, AgileEvolution

TUTORIALS

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

Build Product Backlogs with Test-Driven Thinking—and More
David Hussman, DevJam

Agile Project Failures: Root Causes and Corrective Actions
Jeff Payne, Coveros, Inc.

Specification by Example: Mastering Agile Testing
Nate Oster, CodeSquads, LLC

It's All About Me™: Owning Your Behavior, Improving Your Team
Doc List, Doc List Enterprises

Career Superpowers
James Whittaker, Microsoft

Continuous Delivery: Rapid and Reliable Releases with DevOps
Bob Aiello, CM Best Practices Consulting

JUNE 10–11

THE EXPO

Discover the Top Technologies and Tools All Under One Roof!

TOOLS • SERVICES • TECHNIQUES • DEMOS

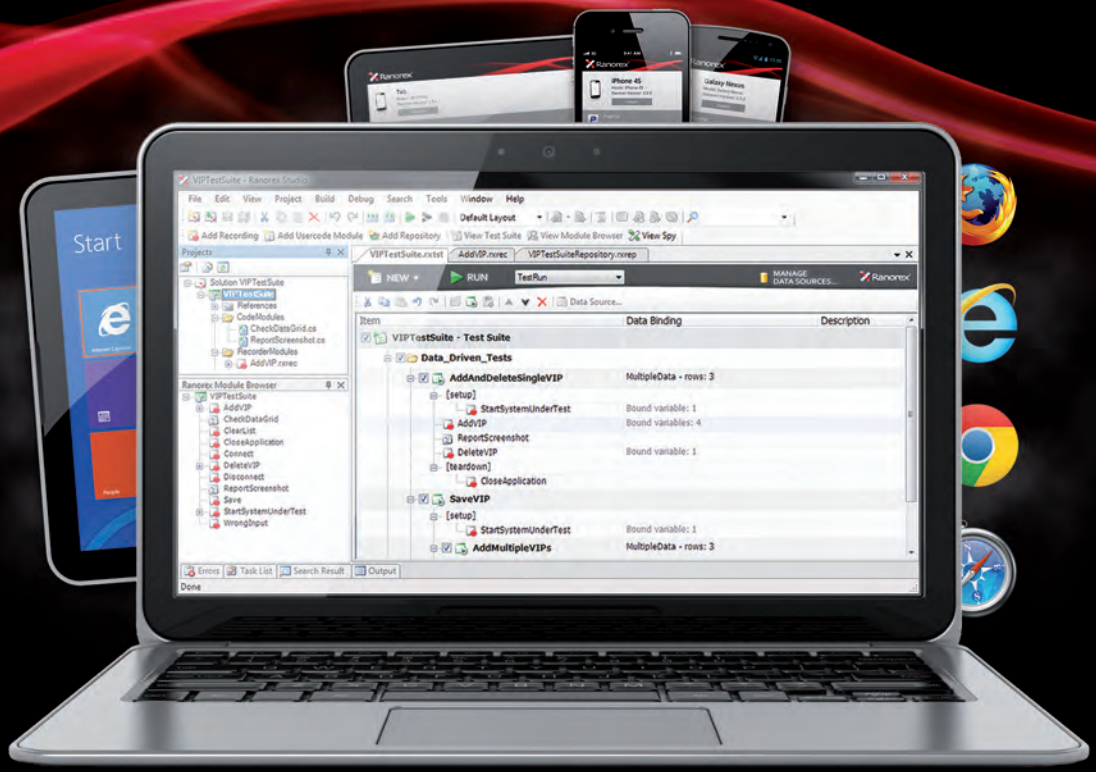
WHO SHOULD ATTEND?

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, and all managers and professionals who are interested in people, processes and the latest best practices in the software development life cycle.

TO REGISTER CALL 888.268.8770 | ADC-BSC-WEST.TECHWELL.COM

Ranorex®

Automated Testing of Desktop. Web. Mobile.



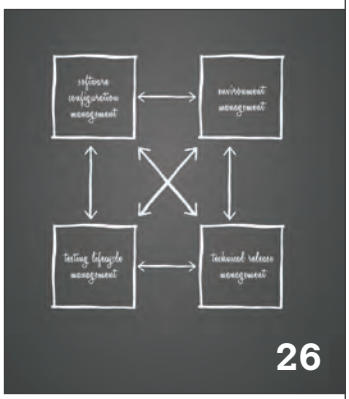
- Robust Automation
- Broad Acceptance
- Seamless Integration
- Quick ROI

Why Use Ranorex
www.ranorex.com/why

Award-winning test automation tools provide seamless testing of a wide range of desktop, web and mobile applications.
Android | iOS | HTML5 | IE | FF | Chrome | Safari | WPF | Flash/Flex | Silverlight | Qt | SAP | .NET | MFC | Delphi | 3rd Party Controls | Java

BETTER SOFTWARE™
A TECHWELL PUBLICATION
Volume 17, Issue 2 • SPRING 2015

CONTENTS



features

columns

in every issue

MARK YOUR CALENDAR

SQE TRAINING

software tester certification

http://www.sqetraining.com/certification

Foundation-Level Certification

April 20–22, 2015
Houston, TX

April 27–29, 2015
San Diego, CA

April 28–30, 2015
Cincinnati, OH

May 3–5, 2015
Orlando, FL

May 5–7, 2015
Raleigh-Durham, NC
Seattle, WA

May 12–14, 2015
St. Louis, MO

May 19–21, 2015
Dallas, TX

June 1– 3, 2015
Chicago, IL

June 7–9, 2015
Las Vegas, NV

June 16–18, 2015
Tampa, FL

June 21–23, 2015
Vancouver, BC, Canada

Advanced-Level Certification

April 20–24, 2015
Atlanta, GA

training weeks

Testing Training Week
http://www.sqetraining.com/trainingweek

April 27–May 1, 2015
San Diego, CA

June 1–5, 2015
Chicago, IL

Agile Training Week
http://www.sqetraining.com/agileweek

May 11–15, 2015
Toronto, ON, Canada

conferences

STAR*EAST*
http://stareast.techwell.com
May 3–8, 2015
Orlando, FL
Gaylord Palms Resort

Agile Development Conference West
http://adcwest.techwell.com
June 7–12, 2015
Las Vegas, NV
Caesars Palace

Better Software Conference West
http://bscwest.techwell.com
June 7–12, 2015
Las Vegas, NV
Caesars Palace

DevOps Conference West
http://devopswest.techwell.com
June 7–12, 2015
Las Vegas, NV
Caesars Palace

STAR*CANADA*
http://starcanada.techwell.com
June 21–25, 2015
Vancouver, BC, Canada
Westin Bayshore

STAR*WEST*
http://starwest.techwell.com
September 27–October 2, 2015
Anaheim, CA
Disneyland Hotel

BETTER SOFTWARE
A TECHWELL PUBLICATION

Publisher
Software Quality Engineering Inc.

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

Sales Coordinator
Alex Dinney

Marketing

Marketing Manager
Cristy Bird

Marketing Assistant
Tessa Costa

SOFTWARE QUALITY ENGINEERING

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

Could This Be the Year of DevOps?

With our upcoming STAR*EAST*, STAR*CANADA*, and Agile Development, Better Software & DevOps West conferences, there's a meteoric adoption of DevOps requiring attention to higher quality processes throughout the software development lifecycle and deployment.

Our authors in this issue of *Better Software* see a different world with development teams attempting to deliver on this new phenomenon of continuous integration and rapid deployment. The reality is that we have a long way to go. There isn't enough testing, internal handoffs tend to be sloppy, and time-intensive manual processes require way too much process and project coordination.

What we need are some practical tips and techniques.

Our cover article from Joe Farah is enlightening for those in the DevOps community who haven't exactly realized the importance of application lifecycle management (ALM) to the testing team. Kim Megahee has many years' experience working with clients on DevOps, and you won't want to miss his practical approach to ensuring that you're taking the right steps dealing with large enterprise deployments. In Technically Speaking, Bob Aiello and Leslie Sachs coauthored one of the most precise definitions of what DevOps really is.

With regard to agile practices, David Hussman's article will make you rethink how you lead your agile teams. Just checking off items from a list is never enough. Wrestling between Scrum and kanban? Stacia Viscardi gives you her perspective from working with a myriad of teams who have found that, although there are benefits to both Scrum and kanban, kanban has a pragmatic appeal to fast-moving software development teams. If you haven't guessed from her article's title, she's a recent kanban convert.

For those of you wondering what a manager looks for when hiring developers, Geoff Perlman gives you a glimpse of exactly what a CEO of a leading software development tools company looks for.

We truly value your feedback. Let us and our authors know what you think of the articles by leaving your comments. I sincerely hope you enjoy this issue!

Ken Whitaker

kwhitaker@sqe.com
Twitter: @Software_Maniac

Contributors

Bob Aiello is a consultant, technical editor for CMCrossroads, and coauthor of *Configuration Management Best Practices: Practical Methods that Work in the Real World*. He has more than twenty-five years' experience as a technical manager in several top New York City financial services firms. Bob served as vice chair of the IEEE 828 Standards working group (CM planning) and is a member of the IEEE Software and Systems Engineering Standards Committee management board. Contact Bob at bob.aiello@ieee.org or linkedin.com/in/bobaiello.

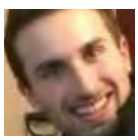
With more than forty years of experience as an information systems professional at commercial and nonprofit organizations, Lee Copeland has held technical and managerial positions in applications development, software testing, and software process improvement. At Software Quality Engineering, Lee has developed and taught numerous training courses on software development and testing issues, and is a sought-after speaker at software conferences worldwide. He is the author of the popular reference book, *A Practitioner's Guide to Software Test Design*. Contact Lee at lcopeland@sqe.com.

President and CEO of Neuma Technology, Joe Farah is a regular contributor to CMCrossroads. Prior to cofounding Neuma in 1990, he was a director of software at Mitel. In the 1970s, Joe developed the Program Library System (PLS), still heavily used by Nortel (Bell-Northern Research), where he worked at the time. He's been a software developer since the late 1960s. Find out more about Neuma at www.neuma.com.

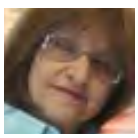
Working with companies of all sizes worldwide, David Hussman teaches and coaches the adoption of agile methods as powerful delivery tools. David often works with leadership groups to pragmatically use agile methods to foster innovation and a competitive business advantage. Prior to working as a full-time coach, he spent years building software in the audio, biometrics, medical, financial, retail, and education sectors. David now leads DevJam, a company composed of agile collaborators. Contact David at david.hussman@devjam.com.

Kim Megahee is a senior DevOps consultant for Zivra LLC. His credentials were won through experience, starting with a large telecommunications company and continuing as he consulted for many Fortune 500 companies through Accenture. His firm specializes in helping enterprises build their DevOps to support faster and more reliable software delivery. He is also a musician and a science fiction writer, who lives near Atlanta GA. Please contact Kim at megahee@zivra.com.

Geoff Perlman is the founder and CEO of Xojo, Inc.,and the creator of Xojo, a crossplatform mobile, desktop, console, and web development tool. Geoff is the company visionary, does quite a bit of the user interface design, and is actively involved in designing features for Xojo. An avid computer user since the age of ten, Geoff taught himself programming and built a career in technology from this experience. Prior to founding Xojo in 1996, he worked in Silicon Valley for a development tools company. Email Geoff at geoff@xojo.com.



A long-time freelancer in the tech industry, Josiah Renaudin is now a web content producer and writer for TechWell, StickyMinds, and *Better Software* magazine. Previously, he wrote for popular video game journalism websites like GameSpot, IGN, and Paste Magazine, where he published reviews, interviews, and long-form features. 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@sqe.com.



Leslie Sachs is a New York state certified school psychologist and the COO of Yellow Spider, Inc. (http://yellowspiderinc.com). She is the coauthor of *Configuration Management Best Practices: Practical Methods that Work in the Real World.* A firm believer in the uniqueness of every individual, she has recently done advanced training with Mel Levine's All Kinds of Minds institute.She may be reached at LeslieASachs@gmail.com or at http://www.linkedin.com/in/lesliesachs.

Stacia Viscardi is an agile coach, Certified Scrum Trainer, and organizational transformation expert devoted to creating energized and excited teams that delight their customers and inspire others. Stacia founded AgileEvolution in 2006 and has helped companies including Cisco Systems, Martha Stewart Living, Primavera, DoubleClick, Google, the Washington Post, and many others find their way to agility. Coauthor of *The Software Project Manager's Bridge to Agility* and author of the recent *Professional ScrumMaster's Handbook*, Stacia has taught agile in seventeen countries. You can reach Stacia at stacia@agileevolution.com.

The State of DevOps Adoption

With all of the emphasis on DevOps and continuous integration with rapid deployment, what shape is the enterprise really in?

by Bob Aiello and Leslie Sachs | bob.aiello@ieee.org and leslieasachs@gmail.com

DevOps is wildly popular and quickly becoming the solution to a wide range of problems. Enterprise agile development would be impossible without DevOps best practices. But the terminology describing this broad transformation puts DevOps at risk of becoming a buzzword that is ill-defined and rendered meaningless due to ambiguity and misunderstanding. While vendors often lead the way to industry best practices, some vendors are guilty of branding their products as being the key to DevOps without actually understanding the principles and practices required to really implement continuous integration, continuous delivery, and the automation of the deployment pipeline. True, DevOps means that your teams work together effectively and efficiently while having the capability to deploy changes as often as necessary and maintaining excellent systems reliability and security. But too many technologists continue to invent definitions of DevOps to suit their own needs without actually reading what recognized industry thought leaders have written based on their professional experience.

DevOps is in need of clarity of purpose and definition along with a greater understanding of its process maturity. The current trend of using DevOps to describe every effective automated procedure is creating much confusion and even dysfunctional behavior. We need to improve our focus on DevOps principles and practices if we are going to realize the potential in this highly effective methodology.

Many practitioners believe DevOps success is a relatively new phenomenon that can be best managed by single push-button deploys and continuous delivery. I disagree with that view, and I also suggest that DevOps has actually been around for quite a while and works equally well with waterfall, agile, and other software methodologies.

The best definition of DevOps is that it is a set of principles and practices that help to improve communication and collaboration between development and operations. DevOps success is best measured by the capability of the team to reliably and securely deploy code as often as necessary and to respond to changes in the system—from application code to large infrastructure upgrades. It also means that the team has the capability to test and verify services in a fast and comprehensive way. It is equally important that the team can monitor the environment, understand all of its dependencies, and respond quickly and effectively to any challenges that arise. These challenges can be most anything; they can be technical or related to organizational dynamics and communications. As many practitioners experience, people-related challenges often outweigh the technical ones.

> "DevOps success is best measured by the capability of the team to reliably and securely deploy code as often as necessary and to respond to changes in the system."

I have seen the power of synergy that develops when stakeholders work together, each having different areas of expertise. I often struggle to get the right people to join the conversation, but, once engaged, they are often the individuals who become the strongest advocates and are eager to collaborate early and often. Having experienced this repeatedly, I am beginning to shape my own views on what DevOps process maturity really looks like.

While many professionals appear to be fixated with push-button deployments, I am far more interested in creating a well-defined, reliable, and verifiable application build, package, and deployment process. Strong communication and sharing of knowledge among developers, testers, operations, data security engineers, and product managers are much more important than being able to push changes out to production immediately without user intervention. Too often, I see developers with extremely efficient continuous integration automation that only works in the developer-controlled environments and frequently cannot be used to support production environments requiring IT controls, as mandated by industry regulatory standards. DevOps takes patience; it is a journey that must be updated as the technology and system itself evolve.

Teaching the team to fish is more important in this case than serving them a great meal.

A mature DevOps high-performance team consists of stakeholders sharing their knowledge and expertise from a cross-section of departments. While organizational silos are often essential for legal and regulatory compliance, intradepartmental thinking needs to be driven out of the organization in favor of extreme teamwork and excellent communication. The mature DevOps team has a strong sense of shared responsibility, optimism, and confidence that never allows finger-pointing and employs systems thinking that requires a strong understanding of the entire application. Success needs to come from the eyes of the customer who provides product and business input that should influence adopting the best DevOps approach.

How do you determine the best DevOps approach?

For any DevOps effort to be successful, it is important to start with an objective assessment, which should be conducted by a DevOps change agent. I usually do this as an independent consultant using the fundamentals in my book, but a well-respected senior member of the technical team also can conduct the assessment. [1] We always begin with interviewing the key stakeholders to discuss existing practices. It is of fundamental importance to drive out any fear so that the information received during the assessment is valid and useful. This early analysis helps set the baseline and identify critical areas that need to be improved. It also helps measure success as DevOps best practices are implemented. DevOps process maturity should be measurable by the team's capability to deploy as often as necessary while ensuring that systems are reliable and secure. **{end}**

Sticky Notes

Click here to read more at StickyMinds.com.

- References

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.
- **Real-time communication:** Communicate real-time directly with the instructor. Ask questions, provide comments, and participate in the class discussions.
- **Peer interaction:** Networking with peers has always been a valuable part of any classroom training. Live virtual training gives you the opportunity to interact with and learn from the other attendees during breakout sessions, course lecture, and Q&A.
- **Convenient schedule:** Course instruction is divided into modules no longer than three hours per day. This schedule makes it easy for you to get the training you need without taking days out of the office and setting aside projects.
- **Small class size:** Live virtual courses are limited to the same small class sizes as our instructor-led training. This provides you with the opportunity for personal interaction with the instructor.

SQE TRAINING
A TECHWELL COMPANY
www.sqetraining.com

ENLIGHTENED SOFTWARE TESTING STRATEGIES

STAR CANADA

JUNE 21–25, 2015
VANCOUVER, BC | CANADA
WESTIN BAYSHORE
#STARCANADA

REGISTER BY APRIL 24, 2015
AND SAVE UP TO $300
GROUPS OF 3+ SAVE EVEN MORE

STARCANADA.TECHWELL.COM

EXPLORE THE CONFERENCE

Choose from 5 days of learning, networking, and more

SUNDAY–MONDAY
Multi-day Training Classes begin

TUESDAY
In-depth half-day Tutorials

WEDNESDAY–THURSDAY
Keynotes, Concurrent Sessions, the Expo, Networking Events, Receptions, and more

KEYNOTES

ILLUMINATING INSIGHT FROM TESTING EXPERTS

STAR CANADA

How We NOW Test Software at Microsoft

Alan Page, Microsoft

Lightning Strikes the Keynotes

Lee Copeland, Software Quality Engineering

Build the Right Product Right: Transitioning Test from Critiquing to Defining

Gerard Meszaros, Independent Consultant

The Next Decade of Agile Software Development and Test

J.B. Rainsberger, JBRAINS.CA

JUST A FEW OF OUR

IN-DEPTH HALF-DAY TUTORIALS

Testers in Value-Driven Product Development

J.B. Rainsberger, JBRAINS.CA

Essential Test Management and Planning

Rick Craig, Software Quality Engineering

Get Full Value from Your Automated Tests

Gerard Meszaros, Independent Consultant

Tips for Expanding Your Testing Toolbox

Alan Page, Microsoft

Getting Things Done: What Testers Do in Agile Sprints

Rob Sabourin, AmiBug.com

Security Testing for Testing Professionals

Gene Gotimer, Coveros, Inc.

JUNE 24–25

THE EXPO

Discover the Top Technologies and Tools All Under One Roof!

TOOLS
SERVICES
TECHNIQUES
DEMOS

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

TO REGISTER CALL 888.268.8770 | STARCANADA.TECHWELL.COM

Jeffery Payne

Years in Industry: **25**
Email: **jeff.payne@coveros.com**

Interviewed by: **Josiah Renaudin**
Email: **jrenaudin@sqe.com**

"If your application is an online banking app, obviously security is going to be of utmost importance. If it's Angry Birds, security is probably not something that you're going to spend a lot of time with."

"It's very difficult to put out an application that's going to work on many, many, many different devices and across many, many different versions of those devices. Compatibility and interoperability is huge."

"I can only test so far and so often, and I can only test certain use cases, if you will, and against certain uses of the product."

"You're starting to see some [mobile testing] tools out there that are interesting and useful. Some of the existing security tools and quality tools are starting to incorporate or provide mechanisms for you to do more automated testing on the mobile device itself."

"A traditional testing approach, where you just focus on requirements and test the functionality of the device, is not going to work in this space. You have got to look at the non-functional requirements, performance, security, usability, configuration, and compatibility."

"I think there's a lot of promise in cloud-based testing to help lessen the burden of dealing with multiple devices and configurations and operating systems, things like that, without having to go buy all that stuff yourself."

"If anybody believes that Apple or Google are adequately assessing the security of all these applications that are showing up, they're fooling themselves."

> "It's real simple. We've got these devices—whether they're phones or tablets or phablets or whatever they are today—that you're using, and unfortunately they're not very safe. They're not very secure."

STICKYMINDS.COM™

For the full interview, visit
https://well.tc/IWAE17_2

THINKSTOCKPHOTOS.COM

Application lifecycle management (ALM) provides beginning-to-end management of your software development project by providing tools to assist with communication, process, and data.

The product team is composed of various project team with members of the test team being key participants in any successful development project. Testers perform a number of important functions, including product verification of requirements, quality level assessment, standards validation, and feedback about process change.

As with any software development team, the test team benefits when automation can be used to help eliminate human error and reduce repetitive tasks. Although test tools have come a long way in supporting automation, the value of testing can be significantly improved when the testing function is integrated with ALM. For example, ensuring requirements have been addressed properly implies that there must be a proper level of integration between requirements management and test case management.

ALM spans team boundaries, allowing every team to provide more value as an active part of the entire project process. Combining an ALM tool with the appropriate process enables clear communication among development teams.

Some key elements of most modern ALM tools can provide even better benefits. Sharing a common data repository, management UI, process engine, and management philosophy provides value to every team. As a result, any improvements made to process and tool infrastructure are more easily justified.

The Importance of Test Case Management

The test team is usually responsible for creating a repository of test cases that verifies product functionality. It must be able to run and record the results of test cases against the product in its various operating environments. Ultimately, it is the recommendation of the test team that determines whether a product is ready to be released.

It used to be that most of a test team's resources were consumed in actual functional testing. Now there is more focus on test automation, allowing more testing resources to be used for test case development, automation, and process improvement.

There are a number of primary and secondary test case categories when attempting to marry testing with configuration management. Summarized in table 1, these test case categories show that the test process touches many parts of the lifecycle.

For the most part, the primary test categories are used to introduce new test cases into the repository, while the secondary

Primary test categories	
Black Box	Verify product requirements
White Box	Verify the design (software requirements)
Change	Verify a change package (i.e., a logical change to the software product)
Bug	Reproduce defects and verify that the known defects are fixed
Sanity	Ensure basic sanity of a deliverable and its platform/infrastructure
Stress	Identify response of a system operated near the edge or outside of the product requirement's envelope
Beta	Identify defects resulting from incomplete requirements, unforeseen use cases, and imperfect testing
Secondary test categories	
Regression	Re-application of test cases to ensure all features still work (e.g., black box)
Feature	Run a subset of black box testing focused on a particular feature
Performance	Run a subset of black box testing dealing with real-time issues
Environment	Run a subset of black box testing identifying the effect of a product on the environment (and vice versa)
Validation	Run a subset of black box testing targeted to meeting specific, required standards
Unit	Run a mix of black box and white box testing at a card, module, or subsystem level
Alpha	Initial beta testing is typically performed using an internal test site

Table 1: Test case categories

test categories repurpose (or reorganize) the existing test cases to provide more targeted test coverage. These categories help to identify the scope of the test team function. There is plenty of data that needs to be collected, created, captured, organized, and summarized.

ALM Helps Organize Your Work

Modern ALM tools provide many of the capabilities not only to keep test cases organized but also to retrieve, deliver, and collect test cases for a single task or purpose. These tools also have capabilities that make it easy to compare sets of test cases (or other artifacts) across builds and releases. This, in turn, provides more precise metrics that allow tracking of product quality as well as development team and process quality. For example, when the test case failure rate increases, it would be nice to know that the primary factor was the introduction of new features that still need some work.

In a silo, a test team can perform testing on a function quite well. But expose that silo to "the rest of the farm" and suddenly management functions better. It has up-to-date test status information it needs to make a go/no-go decision. Having the ongoing, live feedback leading up to that decision, management has the ability to take action to preclude a no-go possibility. In my experience, all functions of ALM are improved by the continuous availability of test information.

If you have been through multiple application development lifecycles, you'll undoubtedly agree that testing is not a standalone function. In an ALM environment, each ALM function contributes to the advancement of the testing process. In return, each ALM function benefits from these testing efforts. Consider the following ways that the various test categories interact with the wider range of ALM functions.

The Importance of Requirements Management

Requirements traceability is critical to any successful project. It is important to know that for every requirement there is a valid set of test cases, and every test case verifies a product requirement. The classical requirements traceability matrix is a sparse matrix, which helps to visually assert this traceability. In practice, with thousands of test cases run against a product, the visual benefit is really only useful in teaching about traceability. The ALM tool that spans both requirements management and test case management should supply the ability to query traceability for any subset of the requirements specification, as well as any subset of the test case collection.

How Configuration Management Supports Build and Release Management

More than supporting requirements traceability, it is important to track the history of actually running each test case against each build and release. The shared data repository afforded by a modern ALM tool should provide easy query access to identify which test cases have successfully passed and which have failed during a given stream of release development.

Requirements typically change over time, and the test suite must change accordingly along with the software being tested. Configuration management is used to ensure that the right revisions of requirements, test cases, and software come together to test the builds leading up to a release.

Generally, test cases are either added to or removed from test case configurations. But there are many complex tests that require version and change control as development of the tests themselves evolves. This version and change control capability is already provided by modern ALM tools for software and their requirements.

As well, each configuration has optional components and variants that are part of the software and build configurations. Testing must take these options and variants into consideration. This is data that is tracked in any modern ALM tool and can be used to aggregate test results to assess the product quality for various components and variants.

Continuous integration requires a certain level of automated testing. The build and test functions of an ALM tool can help ensure that productive testing is automatically performed on each system build. This requires identification of the correct environment data setup followed by running of automated tests (which should not be too time consuming).

Typically, the data associated with a test case needs a few modifications to provide higher level benefits. A simple data field that rates the automation level and running time of a test case or test suite can be used to help select the appropriate set of automated tests to be run against each build.

ALM tools can show the progression over time of test case automation. These tools can estimate expected test times—not only for automated test runs but also for any test session, based on which test cases are selected and which features need to be tested.

Problem Tracking Is a Must-Have

Problem tracking (also known as defect and issue tracking) is another ALM function tightly integrated with test case management. In practice, each problem should have one or more test cases that are used to verify the existence of the problem in existing builds and to ensure the problem has been eliminated in more recent builds. Such test cases should be derived from the problem description as long as the problem has been described in such a way that it is easy to reproduce. Linking the test case to a problem report helps to ensure that all problems have sufficient test case coverage. Such a link also helps to automate the running of a test suite whose purpose is solely to ensure problem reports have been properly fixed.

The ALM tool should support queries such as "When was this problem fixed?" "Is this problem present in build xyz?" and so on. Ideally, a successful running of the test case should automatically advance the state of the problem to the fixed state.

How Feature Tracking and Project Management Fit into ALM

A suite of tests is created based on a feature description document or requirements analysis document. It is important that test cases relate directly to the feature it has been created

to test. This allows easy generation of feature-based test runs.

Success rates can be fed back into the ALM tool to help identify the actual completion level of the feature, at least from a functional perspective. Similarly, there are white box test suites derived from design documents that help test the design of an API, a module, or even a subsystem. Similar techniques can be used to help assess design-level completeness.

This feedback helps improve project management accuracy. If a project manager asks an inexperienced developer about completion status for a task, the response is something like, "Eighty percent complete with 20 percent left to do." It has been my experience that the last 20 percent takes more than 20 percent of the estimated time. Tracking of test case feedback versus developer feedback also provides valuable feedback to the software developer.

Project management is critical to test team planning. When PM is integrated across all ALM functions, resource allocation and schedule bottlenecks become easier to manage more effectively.

ALM Benefits Test Team Value

An ALM-wide approach provides commonality of tools and processes, as well as product-wide visibility of what is really going on throughout the software development lifecycle.

A good ALM tool capability tracks test sessions, allowing simple metrics to be developed for improving the test team's productivity over time. Some ALM tools have code search capabilities that can be used by the test team for searching through test cases. Finally, most ALM tools can organize and report hierarchically. This capability is especially helpful across a changing landscape of new product releases, variant products, and evolving test case content.

The intangible value that results from increased team-wide communication of all ALM information is not to be underestimated. The test team feels more a part of the product team. The rest of the product team is awakened to the various roles of the test team. And as they become more aware, there are efforts to improve the process. Developers see the benefit of formally cataloguing their own tests and passing them on to the test team, both for black box and white box testing. Configuration management personnel identify the need for developers to annotate software updates with the developer test cases used to test those updates, along with test results. The entire team is educated on how to define problem reports so that each problem is easily tested.

It is important to identify ALM tools that provide the capabilities allowing easy and ongoing process customization, shared and extensible data schema, and customization of the UI on a role-specific basis. Tools must be responsive, add noticeable value to each user, and support ALM-wide automation. Once these basics are covered, there is no end to the advances that can be achieved to improve the value of the test team and the entire product team. **{end}**

farah@neuma.com

Curious About DevOps? Start Here!

We can trace the beginnings of the DevOps movement back to a Belgian named Patrick Dubois. In 2007, Patrick lamented that the two worlds of development and operations seemed miles away from each other and there were conflicts everywhere. He observed that development and operations teams tended to fall into different parts of a company's organizational structure (usually with different managers and competing corporate politics) and often worked at different geographic locations. Since then the DevOps movement has gained global momentum and has become a lightning rod for people who have something to say about how IT is—or should be—running.

The movement found traction online through social media and discussion boards. DevOps is clearly touching a nerve within the industry, likely because DevOps is from practitioners, by practitioners; it's not a product, specification, or job title. DevOps is an experience-based movement about cooperation and collaboration.

In response to the call for more information and resources about DevOps, TechWell is introducing the inaugural DevOps Conference West from June 7–12 at Caesars Palace in Las Vegas. DevOps Conference West will accompany the fifth annual collocated Agile Development & Better Software West conferences, the premier event for software professionals. This year's program is even more robust, bringing all aspects of the software development lifecycle to the forefront.

DevOps Conference West features industry practitioners passionate about the DevOps movement and focuses on topics like:

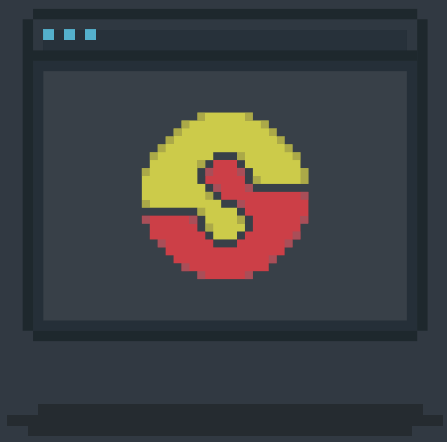
Why DevOps Changes Everything—Keynote presenter Jeffery Payne talks about what steps need to be taken to successfully achieve a DevOps process while avoiding the pitfalls. He'll also leave the audience with some take-home ideas about how to leverage DevOps to advance their careers.

Continuous Delivery: Rapid and Reliable Releases with DevOps—Bob Aiello explains how to implement DevOps using industry standards and frameworks in both agile and nonagile environments, focusing on automated deployment frameworks that quickly deliver value to the business.

A DevOps Journey: Leading the Transformation at IBM—Debbie Edwards describes the journey she went through leading the DevOps transformation at IBM. She will share her experiences, the best practices she discovered, what techniques she used, and how she recommends a software development team get started on the DevOps journey.

We hope your DevOps journey brings you straight to DevOps Conference West. See you in June!

MARTIAL ARTS
HAS BRUCE LEE.



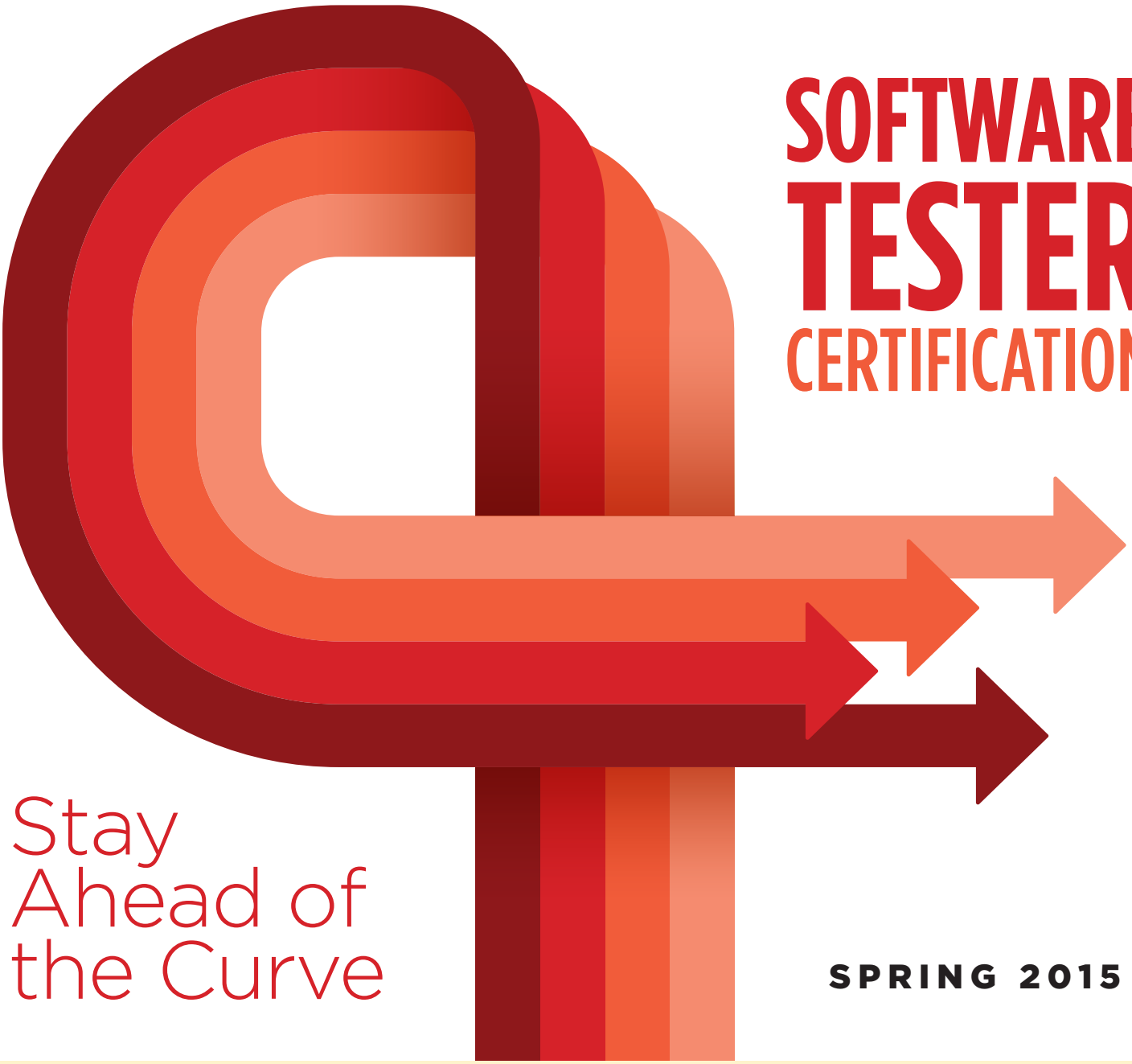
AUTOMATED TESTING
HAS SAUCE LABS.

Maybe you can't do a one-fingered push-up, but you can master speed and scale with Sauce Labs. Optimized for the continuous integration and delivery workflows of today and tomorrow, our reliable, secure cloud enables you to run your builds in parallel, so you can get to market faster without sacrificing coverage.

Try it for free at saucelabs.com and see
why these companies trust Sauce Labs.

Dropbox salesforce YAHOO! PayPal mozilla VISA

sauce LABS



SOFTWARE TESTER CERTIFICATION

Stay Ahead of the Curve

SPRING 2015

SPRING 2015 SCHEDULE

Houston, TX	April 20–22
Atlanta, GA (Advanced Certification Week)	April 20–24
Atlanta, GA	April 21–23
San Diego, CA	April 27–29
Cincinnati, OH	April 28–30
Orlando, FL	May 3–5
Raleigh–Durham, NC	May 5–7
Seattle, WA	May 5–7
St. Louis, MO	May 12–14
Dallas, TX	May 19–21
Chicago, IL	June 1–3
Las Vegas, NV	June 7–9
Tampa, FL	June 16–18
Vancouver, BC	June 21–23

ISTQB Accredited Training Provider

PMI Project Management Institute R.E.P.

Earn up to 37.5 PDUs

- **Basics of testing**—goals and limits, risk analysis, prioritizing, and completion criteria
- **Testing in software development**—unit, integration, system, acceptance, and regression testing
- **Test management**—strategies and planning, roles and responsibilities, defect tracking, and test deliverables

WWW.SQETRAINING.COM/CERTIFICATION

SQE TRAINING
A TECHWELL COMPANY

THINKSTOCKPHOTOS.COM

In early 2000 B.A.E. (Before the Agile Era), I was a technical lead of a small and hardworking software development team. We were trying to wield the newly forming Java enterprise tools following a vague resemblance of the unified process. The players were sharp and skilled, but we struggled to get things done—what the team called *GTD*.

Our continuous battles were new to me. In my previous job our team was successful at creating products and rarely stopped to question why. Knowing we needed to dramatically improve, I started searching for tools that weren't part of the unified path. I found many people exploring lightweight processes and eventually stumbled on *Extreme Programming Explained* by Kent Beck. [1]

The book was short, concise, and applicable to the problems we faced. It contained concrete practices we could immediately apply, and it reminded me of past success. Applying Extreme Programming (XP) values, principles, and practices added sanity to the team. Continuous integration helped us learn faster, and stories started essential conversations. XP helped us find a groove and challenged us to embrace the impact of our choices.

Bonding around Problems

While XP's impact was ubiquitous, Scrum had become the synonym for agile. XP threw down a gauntlet that was too great for many of us modern-day geeks. For those of us who took XP to heart, we found better ways to build more of what was really needed and less of what was not. As a result of applying agile practices, delivery problems surfaced sooner and resulted in our team's bonding around responsible engineering, which led to frequently producing a working system with fewer broken parts.

Like XP, Scrum challenged people to be more open about what they were actually producing. Scrum teams also worked in short cycles with daily meetings. While XP used story cards to spark discussions with customers, Scrum introduced the idea of a product backlog and a product owner. Scrum's product backlogs were a nice complement to XP story cards.

From Process to Product

Practicing XP encouraged the team to produce more by adding less code and cleaning up smelly code with refactoring. The teams I coached were consistently producing working software, and soon I was coaching across teams, domains, countries, and cultures. As I helped teams and programs produce more robust software, my inner skeptic began questioning the real value of what we were producing. Conversations and interactions with testers, designers, and product managers challenged my frame of reference.

Listening to their experiences, observations, and challenges exposed that the piles of unit tests being run for each build and story points we were accumulating did not always result in a better product or user experience. Stakeholders, who were not part of the team, were also questioning the impact.

Their concerns often cut across teams, stories, and iterations. Two common concerns were regression testing and user experiences. Neither was always called out across the backlog items, so both were overlooked during planning that was overly focused on what could be accomplished in the next iteration. XP teams that focused on incremental building were less likely to lose sight of cross-cutting concerns. Cross-cutting is a term that came out of the aspect-oriented programming community addressing essential concepts not contained within objects, but instead cutting across objects during runtime.

Fast forward to today (as shown in figure 1) and you'll often find cross-cutting concerns are merely amplified by consistent rate market changes and evolving technology.

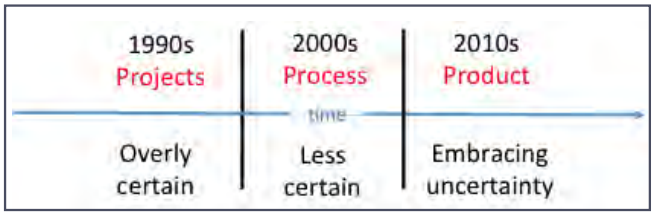


Figure 1: From project to product

Refactoring Your Backlogs

Reimagine the 1990s as the decade of the project, when many of them failed to be delivered or delivered buggy solutions. Next, consider the 2000s as the decade of the process, where delivery and quality improved. Now, imagine the 2010s as the decade of the product, where delivery is less of an obstacle and delivering the right thing is the new challenge.

With that perspective, the importance of backlogs that are more than a to-do list becomes essential, and the focus of iterations moves beyond delivery and toward validation tools. Now, take a moment to assess your process. Are you learning to build the right thing, or are you more focused on completing all the work items for the sprint?

The large number of teams I see with backlogs full of work items speaks to the tendency to fall into doing over learning. Refactoring your backlog to be more of a product learning guide can be done by reviving storytelling and replacing work items with story maps and customer journeys.

Story Writing and Storytelling

A wise friend recently reminded me that prior to user stories, storytelling started with story cards. Story cards are contracts for conversations that promote story-

telling. I am fond of saying that if you don't have a good story to tell, writing it down does not solve the problem. That is where discovery comes in.

Many teams that struggle to deliver usable products have intentionally killed discovery with short planning sessions where storytelling is replaced with sentences like "I am working on 5862." The stories are missing, and the work items are not storytelling starters.

Conversely, teams producing great products usually have storytelling in their DNA. People speak in tests, including phrases beginning with "For example" or "If she does x then she should see y." Some of the best stories come from people connected to the customers and the market. It turns out to be easier to tell stories when you interact with users in the wild or field user frustrations as part of a support call.

Storytelling skills also come from playing the part of the customer and trying to empathize and emulate their experiences, which often cut across teams and stories in the form of workflows and regression tests. If your iterations focus on simply getting work done, you may lose interest from the people who interact with or care deeply about the user's experiences.

Story Splitting versus Customer Journeys

Some of the best conversation starter stories can be quite lengthy. Decomposing them into multiple smaller stories helps promote flow but sometimes leads to focusing too much on size over context, producing a backlog of small stories that don't easily roll up into product or customer validation tools.

Alternatively, story mapping emphasizes context and interactions. Story maps organize a collection of stories in a visualization that should start meaningful conversations. A key example is the ability to select customer journeys that represent the routes you expect and want the team to follow.

If your product is a collection of experiences, the story map allows you to choose different paths to explore instead of prioritizing stories from one to one hundred. This challenges you to decide who you want to take where. Will you go after the person who wants a relaxing walk in the flat part of the woods or the person who seeks an exercise route in the wooded hillside?

A Quick Guide to Story Mapping

Starting with examples is a simple way to build maps. In a sort of test-driven way, start by asking, "What are some examples of how he might do x?" Try to start with an obvious example. It should be what you would expect your target user might do with your product.

Next, create more examples that range in depth of complexity. Start with the obvious example and walk through your customer's experiences. Keep walking the various experiences, exploring options, issues, and various user interactions. Each story you add to the map provides another option you can select as your best guess around what is really needed. Mapping from examples shows you what you don't know sooner. If you can't think of many examples, reconsider your level of certainty and shift your focus from delivery to discovery.

From Doing to Learning

Where process has risen above product, I often hear a lot of talk about teams doing Scrum or teams doing agile. In my observation, doing is a simple verb that we associate with discrete tasks that are completed without much thought. Examples include "doing the dishes" or "doing the laundry."

Compare this to scientists working on ambiguous challenges. You're not likely to hear someone say, "I'm doing chemistry" or "I'm doing physics." Unlike laundry, science and product development are filled with uncertainty and nonlinear experiences. They demand learning over simply doing and call for a readiness to be wrong as well as be done.

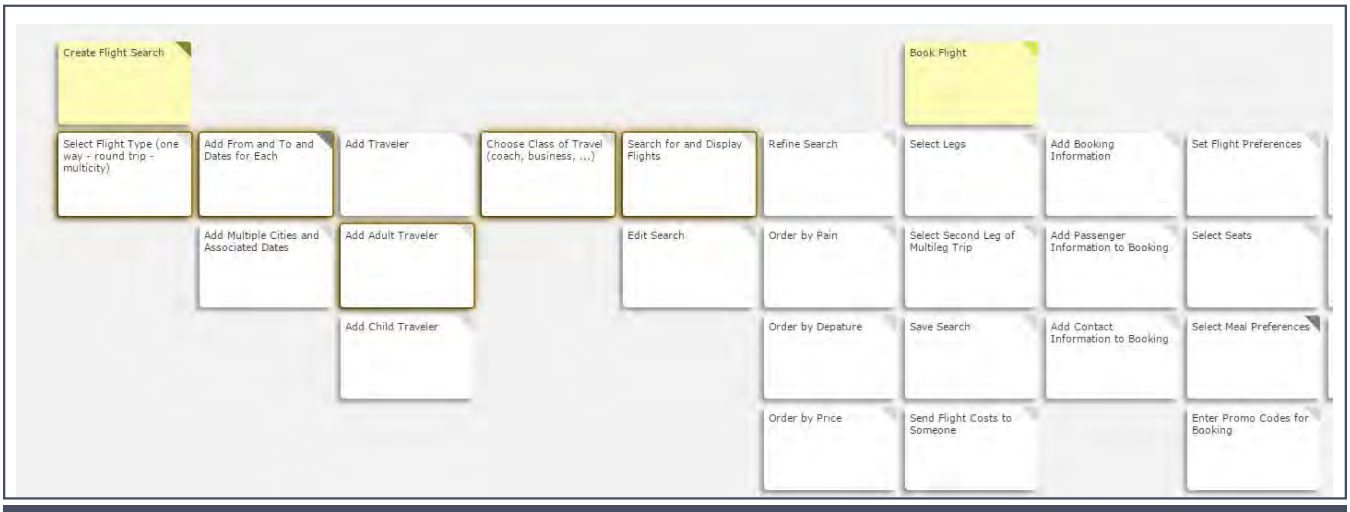


Figure 2: Story map that includes a customer journey

Planning to Discover and Learning from Delivery

With a willingness to be wrong and a readiness to learn, what if we restructure planning as a forum for discovery, where sizing is just one part of the discussion? XP offered this up a bit in the idea of the planning game. This aptly chosen name nicely marries humor, discovery, uncertainty, and winning.

Riffing off the idea of a game, the lead-up to planning might be, "How do we know we are ready to play?" Being ready to play does not mean having all the answers. Games are dynamic, and not all the answers can be known. Rather, being ready means addressing what is known so more energy can be saved for dealing with what is unknown.

Backlogs that are to-do lists refer to a high degree of certainty that getting to done is all that's needed. Working off maps and customer journeys promotes the importance of learning what you don't know while validating some of your other ideas. "Taking customers where you want them to go" over simply getting work done is an essential step in shifting from a to-do list to product learning.

Learning over Sprinting and Validation over Completion

Assuming you're on board with journeys, you're ready to put learning over delivering. Your shift starts by selecting journeys from the map, sizing those stories, and then examining where they land on your iteration boundaries. You might find that this shift helps you more appropriately view iterations as fixed cycles for learning. The other benefit you'll see is the ability to promote cross-cutting learning as a journey along with more traditional story validation in the form of acceptance tests.

Shifting toward validating completed user experiences when they are done may move you to transition from a strong focus on XP and Scrum toward a flow-based style like kanban. Regarding process, a constant theme I hear is that agile teams are focused on building a minimum viable product, which usually means getting small things done fast. Targeting a minimum viable product should be about learning from delivery, and journeys are a concrete tool for validated learning and achieving minimum viability.

Rather than pondering instituting learning over delivery and journeys over work item approaches, you need to consider product design cadences and connecting communities.

Small Is Not Always Better

Many agile teams practice backlog grooming to help right-size stories for iterations. Unfortunately, many grooming sessions split stories at the cost of losing the connection with other stories. An alternative idea is to create a product discovery cadence that feeds a product delivery cadence.

Instead of grooming, establish a cadence of short sessions where product teams explore maps and journeys. This allows for sizing and splitting that is context-based and meets the goals for product backlog grooming.

Backlogs based in maps and journeys provide the cross-cutting content missing from to-do lists. To ground this in a shared experience, imagine you are shopping online. Your experience most likely cross-cuts the many teams that work on the site. As a shopper, you may not care about the teams, their sprints, or their stories—you simply want to have a better shopping experience.

When software teams who created a shopping website meet to review what is needed to be accomplished, would you prefer that they review sprint work completed, or would you prefer they discuss better experiences that benefit you, the user? **{end}**

david.hussman@devjam.com

Never

compromise on quality.

Beyondsoft one-stop testing solution
for all your qualification needs.

With more than 18 years of test development and automation experience, Beyondsoft Benchmark-Driven Testing is the best industry solution to get high quality test results.

Download our white paper on key testing methods that can help you get better and faster test results at **offers.beyondsoft.com/testing**

Contact us:
1-877-896-5859
info@beyondsoft.com
beyondsoft.com

BEYONDSOFT
beyond your expectations

IT Management and Services • Product Engineering • Testing • Mobility • Ecommerce • Cloud Computing

WORLD QUALITY REPORT 2014-15

SIXTH EDITION

YOUR ULTIMATE GUIDE TO THE

Enterprise Application Quality and Testing Practice

- What is the percentage of budget allocated for the testing function?
- Adoption of cloud testing: Hype or reality?
- Does your organization currently test mobile applications and devices?
- How are testing functions adapting to the changing demands of their business?
- How does digital transformation affect organizations' testing practices?

The answers to above queries can be found in the *World Quality Report* 2014-15!

The *World Quality Report*, jointly published by Capgemini, Sogeti and HP, is established as the largest global research study and expert commentary on enterprise application quality and testing practices. The *World Quality Report 2014-15* (sixth edition) is based on market research carried out via telephone interviews with 1,543 senior IT executives across 25 countries.

To know more about Capgemini Testing visit: **www.capgemini.com/testing-services**

Top trends highlighted in the report for North American companies are:

- Mobile testing is rapidly gaining momentum – only one in 10 research participants states that they don't engage in mobile testing.
- Only 16% of North American companies report having no interest in creating Testing Centers of Excellence (TCOE) – a steep decline from 26% in 2013 indicating growing Quality Assurance and Testing maturity levels and a strong drive towards consistency of quality process.
- A growing trend among North American companies is to achieve capex reduction by moving application infrastructure to the cloud.

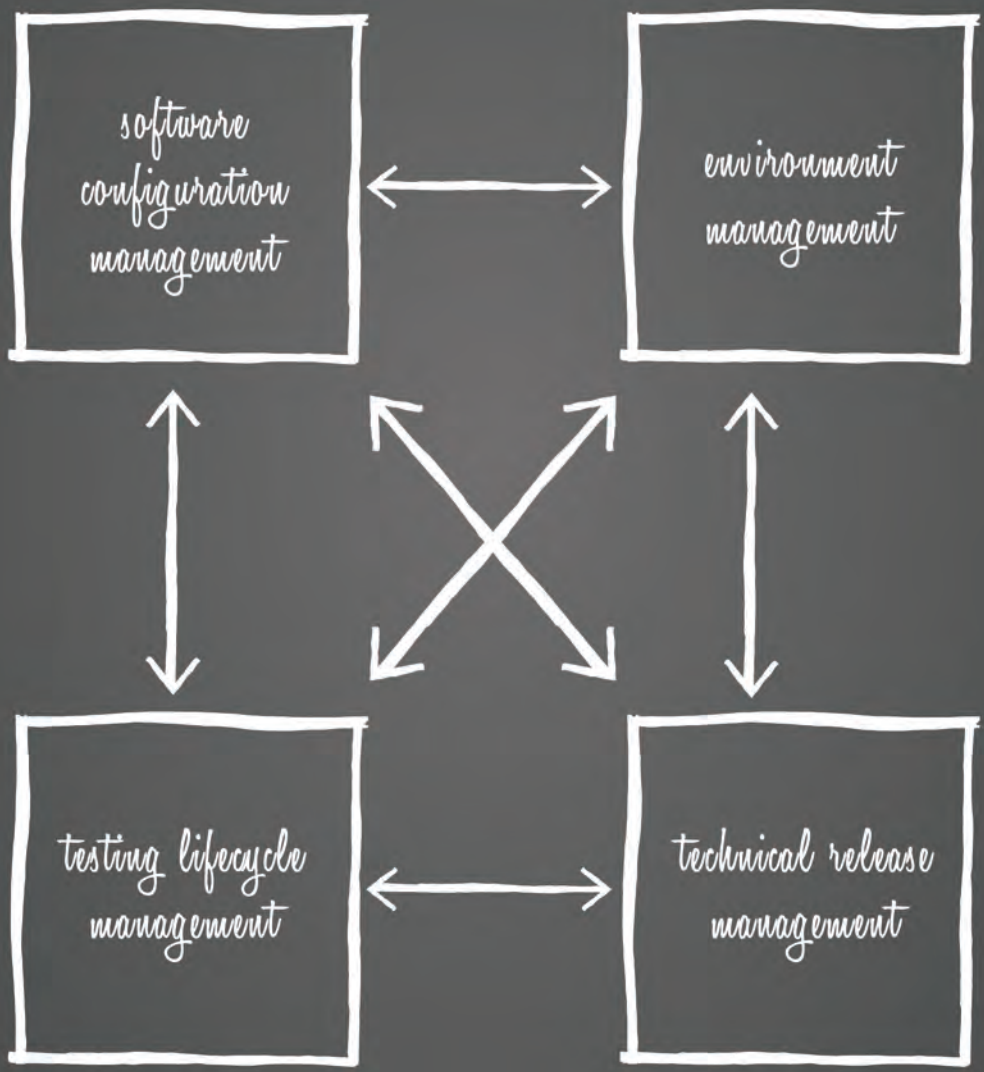
Benchmark your QA and Test team in minutes at **https://benchmark.worldqualityreport.com**

Find out how your organization can adapt to the current trends in Testing. Download the *World Quality Report 2014-15* at **www.worldqualityreport.com**

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

SOGETI

Scaling DevOps at the Enterprise Level



By Kim Megahee

THINKSTOCKPHOTOS.COM

DevOps has been the subject of many articles in the past few years. If you're an IT professional or a corporate executive at a large company and you've been charged with implementing DevOps in your organization, you're probably pretty frustrated about now. I don't blame you. A few years ago, there was a wave of DevOps excitement and, while few people actually understood what it was, everyone raced to get the word prominently displayed in their literature and on their website. I confess that I did it as well. Everyone claims to be in the DevOps business.

To confuse matters, there are quite a few definitions of DevOps out there, and many folks have an almost religious attachment to the one they adhere to. As far as I can tell, the main differences among all these definitions are the speakers' backgrounds and the situations in which they have practiced DevOps activities.

What Is DevOps and the DevOps Space?

My background for DevOps is with large enterprises, where companies have hundreds of systems running apps with millions of lines of code. Allow me to describe what I call the DevOps space. The DevOps space refers to the development phase and the testing phase of the software lifecycle, with a little spillover into production in some cases. The overriding challenge of developing and delivering good software to your users is providing some sort of framework in the DevOps space for getting code created in a "wild west" development phase and migrating it as quickly as possible to a highly structured, stable, and secure production environment.

At the heart of all software development efforts are two activities: creating software and testing software. Every other activity is a support function for those two activities or for moving the product into production.

For the teams I work with, DevOps is simply the set of activities that support development and testing, coupled with a framework for delivering the software to the stable production environment.

DevOps at the enterprise level consists of a number of tools, processes, and highly skilled people. DevOps people, the most important of the three, are individuals who understand the needs of developers and those in operations—individuals who can stand with one foot in each camp and function well in both places.

Can Agile and DevOps Coexist?

While agile projects rely heavily on DevOps, DevOps is not joined at the hip with agile. In my experience, a well-designed DevOps framework functions just fine in most methodologies.

This is not inconsistent with the collaboration aspect of DevOps. After all, collaboration is just working out how to work together to reach a common goal. When you collaborate and write down the result so you can do it again, you create a process. Tools automate processes, and skilled resources operate tools.

As an aside, I've heard that agile doesn't scale well to the enterprise level. I don't believe that's true. The problem is that the conditions required for agile to function as designed can't be met in a lot of enterprise environments. Agile requires the application team have a certain amount of autonomy to function, and most large enterprises have their systems so integrated and codependent that changes to any given system can't be made and deployed without considerable coordination and impact analysis. Most IT managers are not willing to shoulder the risk of giving that much autonomy to a single application team. Agile can be implemented in large organizations but only if the team is empowered with the appropriate amount of autonomy. For this reason, you'll see some very successful agile implementations in large companies, but they will either have a high degree of autonomy or have managed to solve the coordination issues.

DevOps is all about collaboration. It's almost a parable that, in the corporate world, the key to solving any problem is getting the right people together in the same room at the same time. This is what collaboration is all about. In today's enterprises, however, IT systems tend to be a mish-mash of technologies ranging from weeks old to a few decades old, most of them bolted onto each other and codependent—a necessity to solve business problems, adhere to budgetary restrictions, and remain competitive. As a result, it requires a great deal of coordination just to ensure that any software deployment works as planned without breaking something else.

This is not to say that large enterprises cannot benefit from sound DevOps principles. You might not be able to deploy a thousand times a day as some claim, but you can definitely use the principles to significantly improve your ability to create, test, and deploy consistent and stable software solutions to your clients.

The Four Functional Areas of DevOps

We divide the DevOps functions into four main functional areas that are all tightly coupled with each other. Any improvements to one area can have a positive impact on the other areas.

Software configuration management: This refers to the management and control of the changes to the software assets. This typically includes building, deploying, and safeguarding software.

Environment management: This refers to the management of the environments needed to support development and testing, including troubleshooting, provisioning, scheduling, and configuration management.

Testing lifecycle management: This refers to the management of all the aspects of testing support (with some overlap with environment management).

Technical release management: Last, but not least, technical release management is the orchestration of the other functions and includes liaison activities among the project teams, work intake, and DevOps capacity management.

What needs to be done in each area varies depending on the size of the effort and the amount of risk the organization can tolerate. Generally, the bigger or the more critical the application system is, the more important the various function areas

become to the process. Because most applications grow organically and decisions are made to solve problems (rather than to engineer problems out), some organizations progressively move themselves into a situation where they cannot move faster without significantly increasing the risk. This is typically when they start looking for relief in the area of DevOps.

DevOps and the Introduction of Application Kaizen

There is no DevOps silver bullet, but I've found that applying the principles of application kaizen, or Akaizen as I call it, can make a significant impact. Kaizen is synonymous with continuous improvement, and Akaizen is simply applying kaizen principles to DevOps. Each of the four main DevOps functional areas is made up of multiple areas, and they are all tightly connected. The serendipity of it all is that improvements in each of the areas can have positive impacts on the other areas, so much so that the overall quality and speed of software delivery can be dramatically improved.

Some of these areas are foundational. For example, automated builds and deployment can quickly take you where you don't want to go if you haven't first implemented a strong version control system with a well-defined branch and merge strategy. As in every aspect of life, you must be able to crawl before you can walk, run, and fly. Getting all your software into a version control tool that meets your requirements is, therefore, a critical first step, and building a foundation for continuous integration is a worthy goal.

Unfortunately, you cannot just go out and buy yourself some DevOps. You have to build it, one capability at a time. Get the basic, foundational aspects covered first and then build on that foundation. The idea is to break up the work of creating DevOps capability into manageable chunks that can be implemented in three to six months and show real value.

Applying Foundational Activities to the Main Functional Areas

Every enterprise is different. Each has made some attempt to improve its DevOps capabilities. The first step is usually an assessment to determine where the enterprise is before proposing a course of action. But if an enterprise is starting from absolute scratch, there are some foundational activities for each of the main functional areas.

1. Implement an appropriate version control tool and set up an effective branch and merge strategy. Version control is a key part of software configuration management.
2. Create stable and consistent environments by automating their provisioning and configuration activities. Take steps to protect them from unauthorized configuration changes. This is fundamental to supporting the nonproduction environments.
3. Take control of managing test environments and test data in testing lifecycle management.
4. Create a single point of input for work requests so that the workload of DevOps tasks and activities can be measured. Knowing the nature and quantity of the workload drives much of the orchestration and planning of DevOps activities for technical release management.

A skilled DevOps consultant should conduct an assessment to validate the best plan of action. Successful implementation of these foundational activities should have a positive impact on the consistency and stability of your software production and, once implemented, let you move on to the more complex activities your customers are raving about.

Getting the Right Guidance and Experience

As the proverb says, "When you are surrounded by alligators, it is difficult to remember that your objective was to drain the swamp."

To succeed in software development, this is truer than in many other fields. Most IT professionals have good ideas about what needs to be done, but they don't have the bandwidth to do what they need to do—they are just too busy fighting fires in their current situations.

Properly addressing the DevOps foundational areas can be a game changer, and engaging the services of a DevOps expert may be needed.

If you are in an established large enterprise with many interconnected systems that require massive coordination to produce a software release, you might want to be very careful about whom you bring in to help. For example, a resource that built solid DevOps capability in a company whose principal business is a web-oriented sales organization may not have the skills to help you do the same thing in your large enterprise. This doesn't mean they aren't skilled at what they do—it just means their DevOps experience and focus may not be a good fit. Engage with an expert who has been successful with environments similar to yours, and he'll probably be successful.

Conclusion

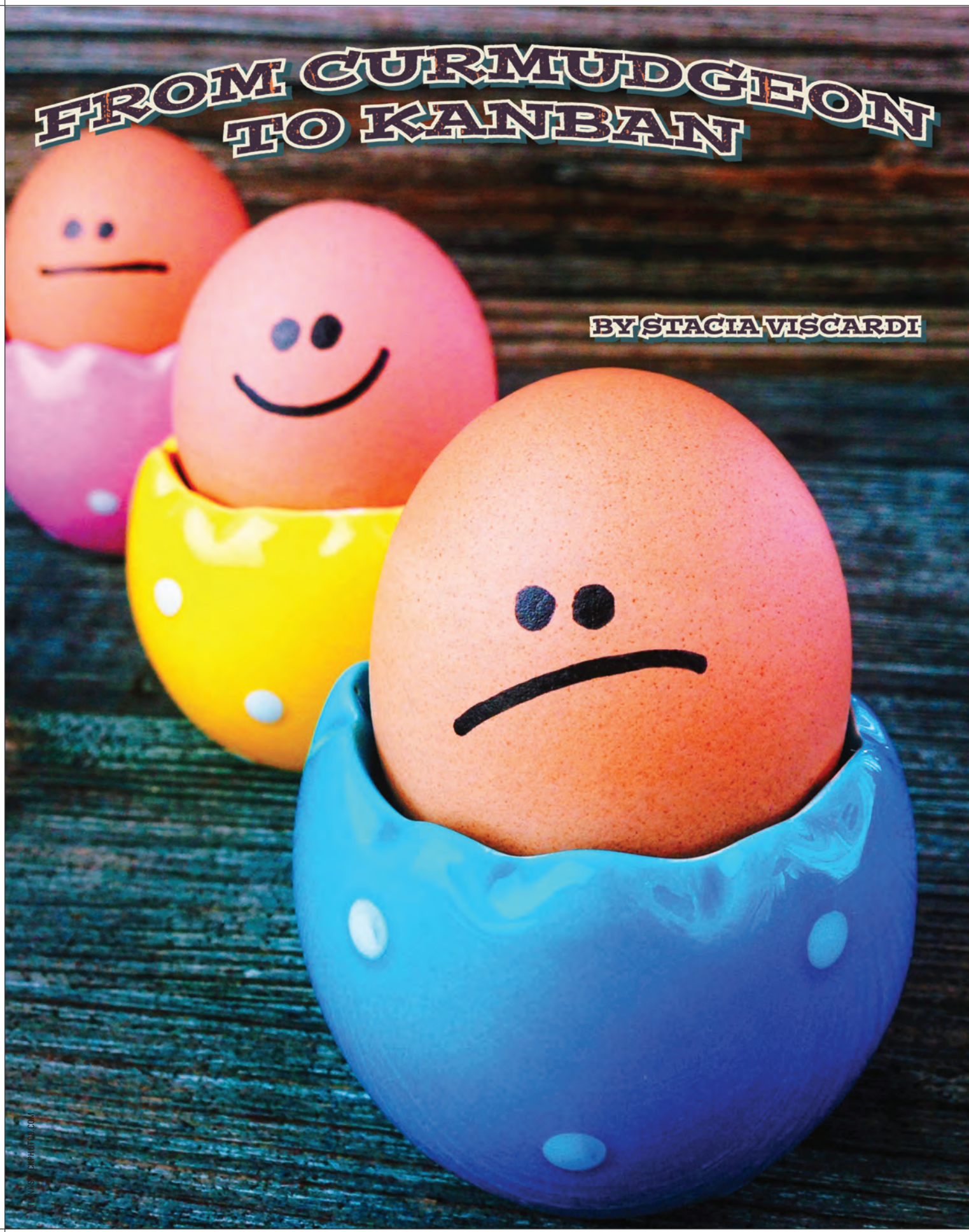
Creating DevOps capability is a journey and not a quick fix. But creating a comprehensive DevOps capability built upon the four functional areas is well worth it. Apply kaizen and start with the basics to build a solid foundation, and find an experienced mentor to guide you along the way. Your customers will benefit from faster, more consistent, and stable software deployments. **{end}**

megahee@zivra.com


BORN to TEST SOFTWARE
STAR WEST
SEPTEMBER 27–OCTOBER 2, 2015
ANAHEIM, CA | DISNEYLAND HOTEL | #STARWEST
REGISTER BY
JULY 31, 2015
AND SAVE UP TO $400
GROUPS OF 3+ SAVE EVEN MORE
STARWEST.TECHWELL.COM

FROM CURMUDGEON TO KANBAN

BY STACIA VISCARDI



Thirteen years ago, Ken Schwaber came to teach our management group about Scrum. I was initially as skeptical as an armadillo crossing a highway. Did Scrum stand for Serious Crud Requested by Upper Management? Was it like water dousing for software teams? I tagged along to training with my peers, notebook in tow, thinking I would just poke some holes in the idea of Scrum, or better yet, see if I could make it fit into my waterfall reality.

Who would have thought that six months later I'd be a burndown-toting ScrumMaster running twelve-team sprint demos? Oh, yes, I drank the Scrum-flavored Kool-Aid. However, underneath my early Scrum parade, I secretly still found solace and comfort in the details of sprint planning. I asked my team to break everything down into four- to sixteen-hour tasks, and I remember wasting my time on an attempt at an algorithm that computed burndowns from actual hours.

> *"A predictable organization does not guess about the future and call it a plan; it develops the capacity to rapidly respond to the future as it unfolds." [1]*
>
> *—Mary Poppendieck*

When I became a Certified Scrum Trainer in 2005, I threw together my first training deck, grabbed Mary Poppendieck's quote, and put it on slide seventeen. It sounded so contradictory, so modern, and even though it made me feel quite rebellious talking about it, I have to admit that it gave me heart palpitations. Even though I had practiced Scrum for a few years, I just couldn't wrap my head around concepts like developing a capacity to respond because my brain was happy with the concept of sprint plans, our team's guess about its one-month future. And for longer futures, we had release plans, too! I was quite comfortable with the planning aspect of Scrum, and curmudgeon-y about any other lighter concept.

While I had certainly heard of it and had read the books, kanban just seemed too "wild west" for me. Just enough and just in time? Sure, but we had to plan it, and it had to wear a coat made of iterations!

After many years of practicing an agile mindset and wrestling with unfolding realities, Mary's words now mean everything to me. Today I'm a personal-kanban-board-using, pull-system-wielding ScrumMaster who finds joy in running iterative experiments. I'm quite comfortable with no estimates at all; in fact, a client recently described me as "living in utopia." Ha, me!

Pulling Away from the Plan-Driven Mindset

My journey into the world of kanban happened through deep reflection upon a series of enlightened moments. The following are a few of the key observations that over time helped me loosen the reins on a push-management mindset to enjoy the ride of a value-driven, pull-system mindset, otherwise known as kanban.

Sprints are two-week vacations in work breakdown structure wonderland. There is safety in sprints. Let's face it: Aren't they really mini-projects? If we reduce the idea of a sprint to its simplest form, we could think of it as a fixed time, fixed scope, and fixed cost approach to delivering a big list in small chunks. This sophomoric way of thinking allowed me to initially adopt Scrum because it felt like the way I traditionally managed projects, but on a smaller scale. Once I got the fact that I was running two-week projects—without the Gantt charts—my pulse settled. I'm surely a ScrumMaster now!

And then I realized that there was much, much more to this.

We like our customer? "Wait a minute! The customer is a team member, too." All of a sudden, scope becomes this twisty, sneaky, creeping, evolving blob. Negotiation abounds and everyone appears satisfied. The team seems to enjoy working with the product owner. What is this new win-win world? My traditional way of managing projects had never resulted in such enthusiastic engagement of product owners and teams, sprint after sprint, in true partnership. So if the team likes it and they're happy and productive, how can I not like it?

How would you like to spend your Monopoly money? Over the years, I've been fascinated by watching teams develop a sense of velocity—the measure of work completed in a sprint. Once a team develops an understanding of its velocity, the nature of the conversations moves away from defining the exact content of a sprint or release to the team's capacity to take on work and how that might extrapolate over time.

Team Awesome knows it has a capacity for one hundred points of work in a sprint. One hundred points of what, exactly? Well, Mr. Customer, whatever you'd like to use those points on. Story points are like Monopoly money—spend them on railroads, hotels, or a new style sheet, report, or cloud capability. As I observed many teams find their velocity, Mary Poppendieck's ideas about planning versus capacity began to make more sense to me. I had begun my transition to accepting a world of unfolding experiments instead of planning everything to perfection.

Deliver stories in small batches. I remember early on, a team ended a sprint with all the stories 90 percent complete but nothing really finished. Somewhere in the hopelessness and despair of zero velocity, my team realized that perhaps it made more sense to finish a couple of stories early in the sprint, followed by the next few stories, and so on. So we began something that resembled kanban within a sprint and never suffered the perils of velocity equals zero again. Better yet, instead of waiting an entire iteration to accept stories, the customer could do so as soon as stories were ready. Some people refer to this as Scrumban but it ultimately doesn't matter what you call it: Delivering in small batches focuses a team on the most important items, boosts throughput, and promotes quality.

Maximize throughput and eliminate rework. A team I worked with a few years ago undertook a yearlong refactoring effort to enable test automation. This was slow going in the beginning, and the business complained that the team wasn't as productive as before. However, six months after the refactoring effort and hundreds of automated tests later, the team improved from delivering partially tested features once a month to fully tested features every day. It was important to

team members to know how much faster and frequently they could deliver high quality features, so they measured cycle time and rework, among other things, to justify the investment to the business. They did this at the onset and throughout the effort. Additionally, the team began keeping tabs on lead time—the time from when a request was initially made to when it was delivered. The team and product owner learned how much upfront planning—a grand portion of it useless—they had previously entertained. They shared this information with the business, which opened up thoughtful discussions about forecasts and budget cycles. By seeing versions of this example play out time and again, I've learned that as we bring throughput into focus, instead of conformance to plan, emphasis on timeboxes may very well fade in favor of more meaningful metrics that help us truly understand success.

Drop the iterations. Does your team need iterations as a useful cadence for business or customer engagement? Or do those interactions occur more organically? If it's the latter, perhaps it makes sense to drop iterations. I'm not recommending that for every scenario, but it might work in yours.

Go ahead, have some fun. Of all my observations over the years, the most striking is the difference in teams that are rigidly controlled and those allowed to self-organize with trust and accountability. The latter are happier, more satisfied, engaged, and committed. They develop opportunities to learn, teach, and proactively solve problems to delight their customers. I love it when people smile and high-five after they've solved a problem together, go home at a decent hour, and relish in the satisfaction of a job well done. A tidy project plan just can't give you that feeling.

Practice, Learn, and Surprise Yourself

I have evolved to think of Scrum not so much as an end to transition to, but rather as something to transition through. Agile, Scrum, Scrumban, kanban—and the list goes on—are not methodologies. They don't represent the end but the means. Teams can use these frameworks to adapt their own best ways of working to suit their individual needs and dynamic challenges. A little dash of Scrum, blended with Extreme Programming, topped off with a pinch of kanban and a dollop of continuous reflection makes a darn good development jambalaya.

The key to learning anything new is to suspend your fear long enough to experiment. There is a scary edge in that place, but it is on this edge, staring down a ten-thousand-foot drop, where you will experience deeper understanding. As an initially reluctant, sophomoric adopter, I now possess an intrinsic lean mindset—a set of values, ethics, and priorities that my old self would have deemed crazy many years ago. **{end}**

stacia@agileevolution.com

Sticky Notes

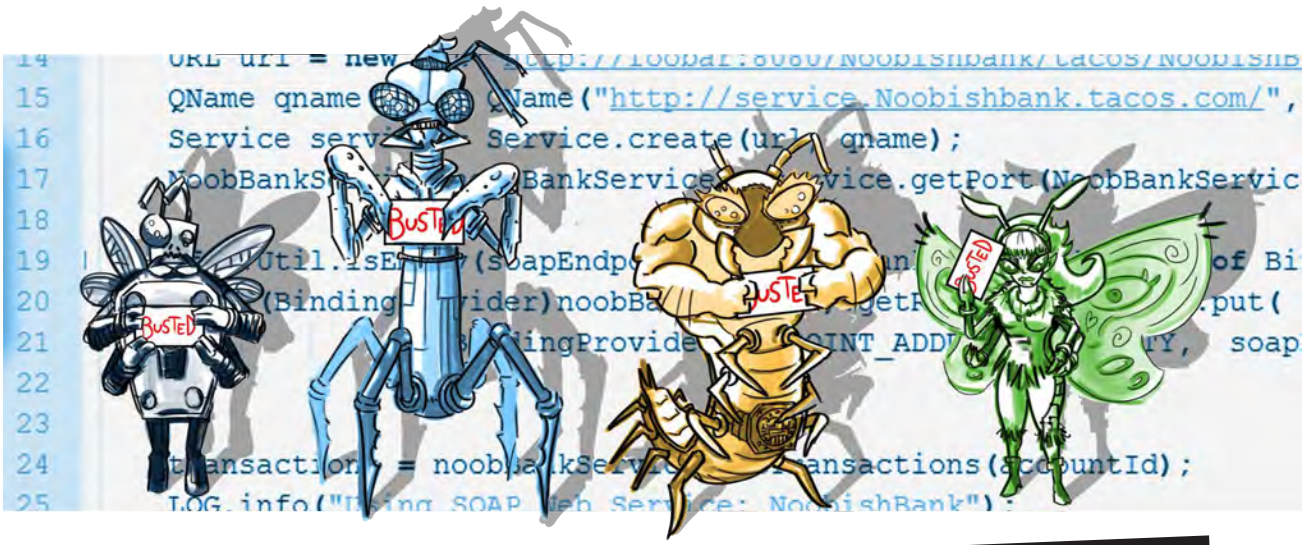
Click here to read more at StickyMinds.com.

- References

PARASOFT®

BUSTED!

Software bugs can't hide from **Parasoft's advanced defect detection and prevention technologies**



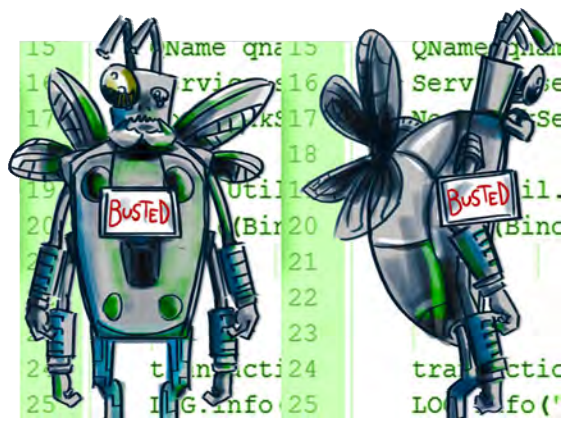
Parasoft software quality technologies, such as **static analysis, unit testing, peer review,** and **coverage analysis** ensure defect prevention for the next generation of applications.

STATIC ANALYSIS

UNIT TESTING

PEER REVIEW

COVERAGE ANALYSIS



Visit **www.parasoft.com/busted** to sign up for a free evaluation.

Stop by the Parasoft booth to learn more about how Parasoft can bust bugs in your code!

AGILE DEVELOPMENT CONFERENCE EAST

DEVOPS CONFERENCE EAST

BETTER SOFTWARE CONFERENCE EAST

Register By September 11
AND SAVE UP TO $400
Groups of 3+ save even more

THREE CONFERENCES
IN ONE LOCATION
REGISTER FOR ONE AND ATTEND SESSIONS FROM ALL THREE

ADC-BSC-EAST.TECHWELL.COM | #BSCADC

PMI® Project Management Institute R.E.P.
PMI® members can earn PDUs at this event

FAQ

expert answers to frequently asked questions

What Are the Key Components of an Effective Test Strategy?

by Lee Copeland
lcopeland@sqe.com

A strategy is not a collection of detailed step-by-step instructions; rather, it defines goals to be achieved, often with limited resources and under conditions of uncertainty.

Quality assurance organizations can definitely benefit from having their own strategies for testing. A test strategy is a high-level plan to achieve specific test objectives, given the uncertainties of the product's quality and within the constraints imposed by the organization. Test objectives vary with each project, usually including finding defects, gaining confidence, defect prevention, and evaluating product usefulness. A test strategy clarifies the major tasks and challenges, and is the set of ideas that guide test design.

James Bach, the inventor of rapid software testing, reminds us that a good strategy is specific, practical, and justifiable. It is specific in that it guides our decisions, practical in that it can actually be performed within the resource constraints we have, and justifiable in that there are logical reasons for the strategy.

On the other hand, a strategy is not a set of test cases, procedures, or data—all of which should have been designed based on a previously defined strategy. The components of an effective strategy typically address the following:

Identification of constraints: Knowing the constraints imposed on our testing allows us to prioritize and then eliminate items from our testing that we will not be able to perform, giving an opportunity to do the most important tests first.

Identification of software functions, risks, and priorities: This allows us to define what we will test and the coverage we expect to achieve.

Selection of risks to be evaluated: Test levels of the type of tests to be run need to be defined, including unit, integration, system, acceptance, usability, security, performance, alpha, and beta.

Each test level is focused on a specific type of risk. Unit tests focus on the risk that the developer did not write correct code; integration tests focus on the risk that the program modules do not play well with each other; performance tests focus on the risk that the system does not perform at the level needed by customers; usability tests focus on the risk that the system is difficult to use; and so on.

A test strategy defines the most important risks and the test levels that will be used in testing the product.

Defining and establishing a testing environment: Testing is typically performed in a separate test environment from production. The test strategy defines the capabilities of the test environment.

Providing a tool for each test: These tools are directly related to the risks and test levels used in product testing.

Prepare for the final exam: This defines how, when, and to whom the information generated by the testing will be made available.

The purpose of testing is to create, organize, and disseminate quality-related information that others can use to make better decisions. A test strategy is necessary to describe how that should be accomplished. **{end}**

Hire the Right Developer

Beyond having great technical skills, a job applicant needs to be a fit in your organization's culture and have a true passion for his craft.

by Geoff Perlman | geoff@xojo.com

When it comes to business success, hiring the right people comes second only to providing a product or service that customers want. No matter how good your product or service is and no matter how revolutionary or disruptive the idea, if you don't have the right people, the thing you are selling (and perhaps your company) is doomed to failure. While hiring the right people is important to all aspects of any business, my company creates software development tools, so I'm going to focus on hiring the right developers.

There are three characteristics I look for when it comes to hiring the right developers: skills, fit, and passion. If you were evaluating a candidate using an A to F grading scale, the right developer to hire needs to get an A in two of these categories and at least a B in the other.

Skills

You need to know clearly what skills are required. If you need someone who is skilled at programming in C++, don't hire someone who knows C but promises he will get up to speed on C++. You don't have time for that. Need a developer with Android experience? Don't hire an iOS developer and expect her to learn Android. Be clear on what skills are required and which ones are not. On the other hand, if the developer has used SVN for source code versioning and you're using Git, that's probably not a big deal. A new hire can pick that up quickly.

> "You need to know clearly what skills are required. If you need someone who is skilled at programming in C++, don't hire someone that knows C."

And what about education? Ask yourself if a computer science degree is really that important. And what about residency issues? If you are considering someone from outside the country, there may be visa issues if he doesn't have an applicable college degree. Other than that, recognize that a lot of great developers are self-taught. If I had two developers who appeared to be equally skilled but one started teaching himself programming when he was ten years old and has no degree while the other guy didn't start learning programming until he went to college for his computer science degree, I'd take the self-taught candidate—especially if he demonstrates initiative and passion for programming.

The bottom line is that you probably don't have the time to train someone. In this fast-moving technology industry, a new hire needs to have the skills to become productive quickly. And the larger your code base, the longer it's going to take before even a skilled and experienced developer is truly productive.

Be clear in your own mind about what skills are required and narrow your search to only those who have the required skills. If that leaves you with only a few candidates, then find a way to get more prospects.

Fit

Even if the right developer has the skills you are looking for, he also has to be a great fit with your team and your organization. If your team uses Scrum, a great developer will need to be comfortable working in an agile and totally transparent way. If your organization insists on performing code reviews, the right developer will not only want to review other developer's code but also should be equally comfortable receiving constructive criticism. On the other hand, if the candidate is more of a loner or objects to daily scrutiny, he simply isn't a good fit.

And what about location? Do you need all your developers in one location, or can they work remotely? Know this up front and make it clear in the job description so you don't waste time with candidates who aren't a good match. At Xojo, most of the developers used to be located in Austin, Texas. Over time, however, we have hired more developers who work from home. We have found that works well for us, and it removes the potential objection to relocation.

Does your team have a coding style? A great candidate will be fine adapting to it.

Finally, does your team require developers to create unit tests for their code? The right developer will already be doing that in her own code.

Passion

In my opinion, passion is the most overlooked and underrated attribute of a great developer. I don't mean just passion for software development—that's a baseline requirement. I'm talking about passion for the product or service he will be working on. Is the developer looking for any programming job that will pay him a decent salary, or does he want to work for you because he loves your product or service? In my experience running Xojo, the developer in the latter category will be far

more productive than the former. He will put in extra hours, come up with new ideas, and demand a higher level of quality because he genuinely cares about what he is doing.

We try to hire customers. Customers are often already very passionate about the product or service or they wouldn't be using it. In our case, we make development tools, so every customer is at least a potential candidate. But even if your customers are not all developers, there will likely be some software developers among them. Find them and you are more likely to find a passionate developer. Although we have successfully hired developers who were not customers, there's a world of difference between those who are and those who are not. I'd take a candidate who is a B for skills or fit but is an A for passion anytime. A passionate developer will work harder to improve her skills and fit in.

Don't Rush It

You're probably anxious to get someone started. I get that. Finding the right developer may take longer than you'd like, but dealing with the wrong one will be far more time-consuming and expensive. Have everyone on the team interview the candidate, then take their evaluation very seriously. Hiring a developer after the team makes it clear he was not the best choice can negatively impact everyone. You will lose the respect of your team and send the message that their opinions are not important to you. This will cost you in lost productivity and potentially in additional turnover (and in our industry, turnover is often very expensive).

Your team can be a great source of potential candidates. Just be careful you don't hire anyone too close to an existing member of the team. When a team member suggests considering one of their friends, ask him how he would feel if that friend had to be let go in the future. If that's going to be awkward, it's probably best not to hire that individual in the first place.

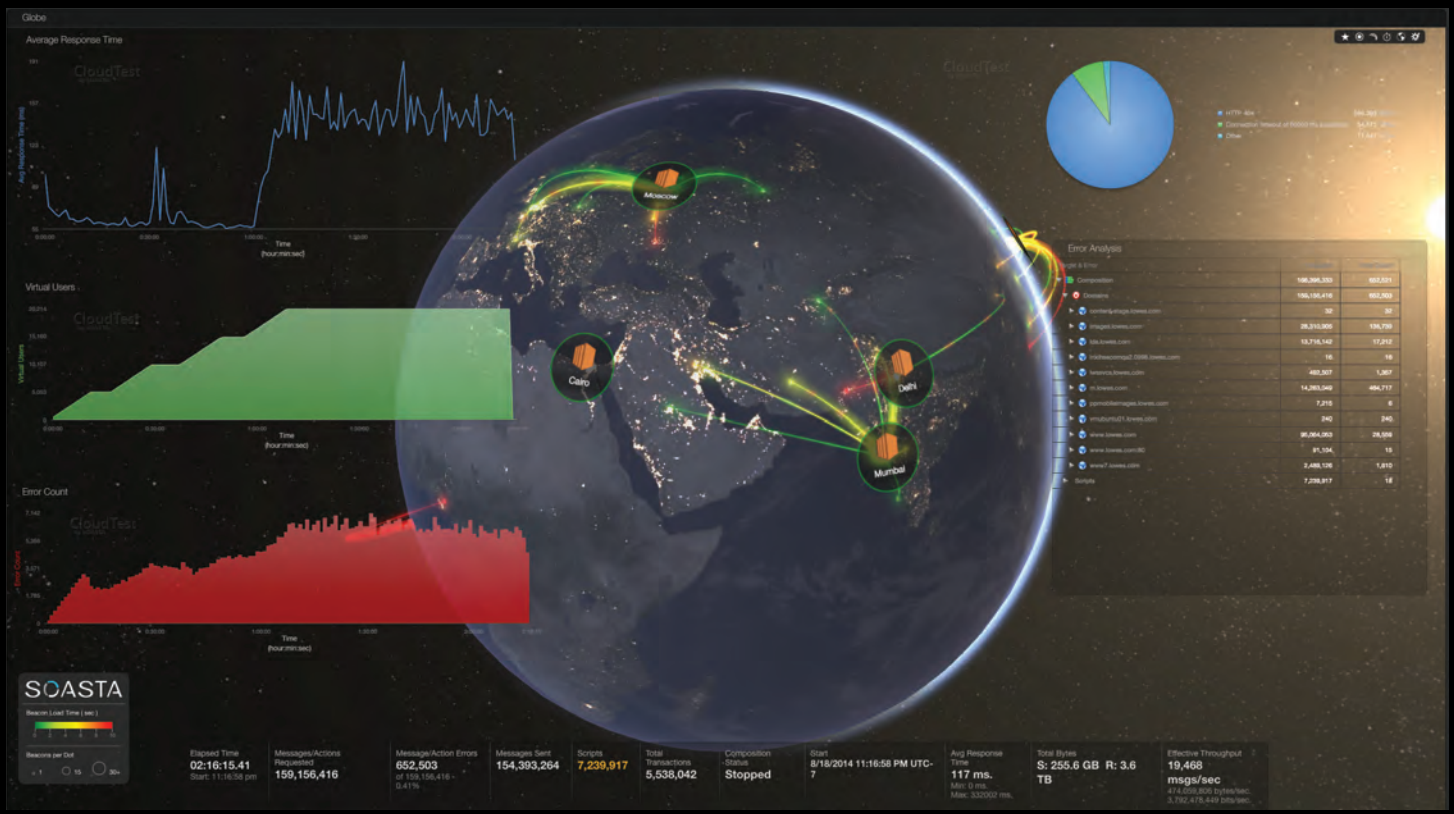
When you find a developer who has the right skills, fits with the team, and is passionate about what you do, the productivity gain—both in terms of the individual developer and the team—will be noticeable. It is better for your business to take the time to find the right developer than hiring the first decent match that comes along. The extra effort required will be worth it. **{end}**

index to advertisers

Better Software (ISSN: 1553-1929) is published four times per year: January, April, June, and September. Print copies can be purchased from MagCloud (http://www.magcloud.com/user/bettersoftware).

SOASTA

Modern apps demand modern testing



Optimize your results with the power of real user data

Monitor Real Users	Measure at Cloud Scale	Optimize Continuously

Learn how: www.soasta.com