

BETTER™ SOFTWARE

A TECHWELL™ PUBLICATION

TEST AUTOMATION
We Must Get It Right

AGILE GOVERNANCE
Value Metrics That Work

HOW TO AVOID UNNECESSARY TECHNICAL DEBT

IN

MOBILE APP
DEVELOPMENT
PROJECTS



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



October 2-7, 2016

Anaheim, CA | Disneyland Hotel

STAR WEST

A TECHWELL EVENT

[HTTPS://WELL.TC/SW16BSM](https://well.tc/sw16bsm)



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:

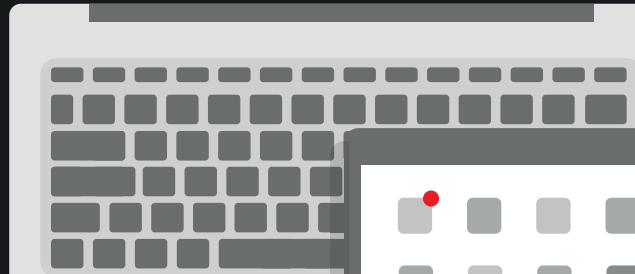
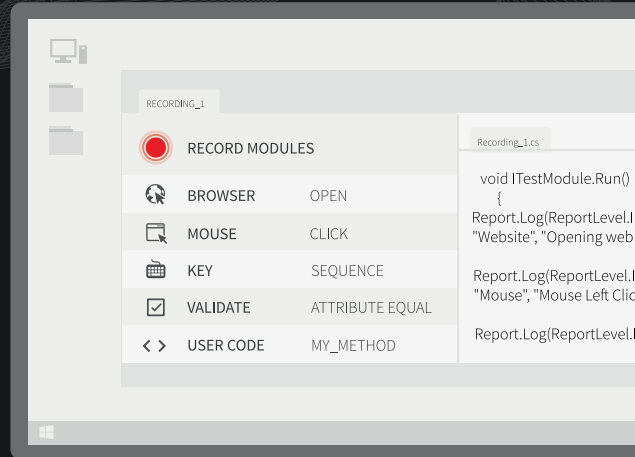
- **Easy course access:** You attend training right from your computer, and communication is handled by a phone conference bridge utilizing Cisco's WebEx technology. That means you can access your training course quickly and easily and participate freely.
- **Live, expert instruction:** See and hear 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.








www.sqetraining.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 HOW TO AVOID UNNECESSARY TECHNICAL DEBT IN MOBILE APP DEVELOPMENT PROJECTS

When task completion falls behind and more work is lined up for later, you've entered the land of technical debt. This is particularly true in mobile app development. Brian Westendorf presents practical advice to avoid this situation.
by Brian Westendorf

18 TEST AUTOMATION: NOW WE HAVE TO GET IT RIGHT

Never afraid to voice his opinion, Paul Gerrard suggests that in digital transformation projects, test automation may be the biggest challenge to success. He argues we have to get automation right this time and to do this, a new way of thinking about testing may be required.
by Paul Gerrard

24 VALUE METRICS FOR AGILE GOVERNANCE

In agile projects, team-level metrics are not useful for planning and monitoring projects across a software development organization. According to Mike Harris, the best value measurement should be based on providing customer value.
by Mike Harris

30 THE BENEFITS OF EARLY PERFORMANCE TESTING

By incorporating performance testing early in a project lifecycle, software projects have a better chance to reach better quality *and* meet customer expectations. Baljeet Bilkhu shows the value of early performance testing.
by Baljeet Bilkhu

columns

7 TECHNICALLY SPEAKING YOUR PROFESSIONAL RESPONSIBILITY FOR SECURITY AND PERFORMANCE

It is Johanna Rothman's belief that security and performance are no longer nonfunctional requirements in modern-day software development. Instead, we must prepare to accommodate security and performance needs in all projects.
by Johanna Rothman

36 THE LAST WORD EMBRACING THE TOP TRENDS IN SOFTWARE TESTING

As user needs change for software apps expanding into IoT, mobile, and the cloud, testing approaches need to change. Shyam Ramanathan discusses eleven of the most important testing trends you should incorporate.
by Shyam Ramanathan

MARK YOUR CALENDAR

training weeks

Testing Training Week

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

April 4–8, 2016

San Diego, CA

June 13–17, 2016

Chicago, IL

August 22–26, 2016

Dallas, TX

September 19–23, 2016

Washington, DC

software tester certification

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

April 12–14, 2016

Nashville, TN

April 26–28, 2016

Washington, DC

May 1–3, 2016

Orlando, FL

May 17–19, 2016

Austin, TX

May 24–26, 2016

Louisville, KY

Omaha, NE

June 5–7, 2016

Las Vegas, NV

conferences

STAREAST

<https://stareast.techwell.com>

May 1–6, 2016

Orlando, FL

Renaissance Orlando at SeaWorld

Agile Dev West

<https://adcwest.techwell.com>

June 5–10, 2016

Las Vegas, NV

Caesars Palace

Better Software West

<https://bscwest.techwell.com>

June 5–10, 2016

Las Vegas, NV

Caesars Palace

DevOps West

<https://devopswest.techwell.com>

June 5–10, 2016

Las Vegas, NV

Caesars Palace

STARWEST

<https://starwest.techwell.com>

October 2–7, 2016

Anaheim, CA

Disneyland Hotel

STARCANADA

<https://starcanada.techwell.com>

October 23–27, 2016

Toronto, ON

Hyatt Regency Toronto



SPRING IS IN THE AIR

Now that winter is turning into spring, it is time to renew our thinking about quality. The spring issue of *Better Software* magazine includes a number of articles that emphasize state-of-the-art practices in testing, metrics, and controlling technical debt on mobile software development projects.

Technical debt—the accumulation of errors in architecture, design, and poorly written code—can negatively impact even the best-laid plans for a software development project. Our cover feature article, “How to Avoid Unnecessary Technical Debt in Mobile App Development Projects” by Brian Westendorf, gives a unique perspective for recognizing and then controlling technical debt.

In the last issue of *Better Software* (Winter 2016), Rajini Padmanaban and Ross Smith introduced us to the merits of gamification as a key ingredient to modern-day learning management systems (LMS). In this issue, Baljeet Bilkhu’s article, “The Benefits of Early Performance Testing,” shows us the benefits of early testing in the creation of LMS software.

If you thought you understood test automation, you’ll want to read Paul Gerrard’s “Test Automation: Now We Have to Get It Right.” Paul does a masterful job of explaining a new model for testing that should help in test automation planning.

In “Value Metrics for Agile Governance,” Mike Harris gives us insightful ways to apply meaningful value metrics in agile project development.

With the industry moving to millions and billions of Internet of Things devices, Johanna Rothman explores why your testing can’t take an application’s security and performance for granted in “Your Professional Responsibility for Security and Performance.”

And those of you who test for a living will want to read Shyam Ramanathan’s column, “Embracing the Top Trends in Software Testing,” in which he identifies software testing’s eleven top trends.

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! And please use social media to spread the word about *Better Software* magazine.

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 Assistant
Tessa Costa

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



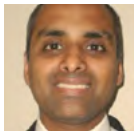
BALJEET BILKHU has worked for more than twenty years in various QA-related roles. Baljeet has been a team lead, manager, program planner, and most recently a senior test strategist at D2L. He actively advocates for more upfront automation and performance testing by mentoring and collaborating with testers in an agile environment. Others in his organization often consult with Baljeet about the workings, best practices, and innuendos of JMeter. Contact Baljeet at baljeet_bilkhu@hotmail.com.



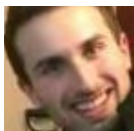
PAUL GERRARD is an internationally renowned, award-winning software engineering consultant, author, and coach. Paul was the host of the UK Test Management Forum and program chair of the 2014 EuroSTAR testing conference. A consultant, coach, mentor, author, webmaster, programmer, tester, conference speaker, rowing coach, and publisher, Paul has conducted consulting assignments in all aspects of software testing and QA. He specializes in test assurance, test strategy, open source implementation, test automation, shift-left approaches, and helping testers become more business-aware and/or more technical. Contact Paul at paul@gerrardconsulting.com.



MIKE HARRIS has more than thirty years of broad management experience in the IT field, including periods in R&D, development, production, business, and academia. An author and speaker on a range of topics related to the value visualization of IT, Mike is considered a thought leader in the software development industry. He is the president and CEO of DCG Software Value and previously held numerous senior executive positions at Fidelity National Information Services, Sanchez Computer Associates, and MasterCard International. Contact Mike at m.harris@softwarevalue.com.



SHYAM RAMANATHAN is the director of software quality assurance, North America, at Virtusa. In the quality assurance world for more than fifteen years, Shyam has experience in test management, planning, design, execution, establishing global delivery teams across geographies, and building test centers of excellence. His responsibilities include business development, project and program management, QA management, partner strategies, technical oversight, and delivering creative solutions. Shyam has run five half marathons and one 25K run, is an avid blogger, and loves reading business books. Contact Shyam at sramanathan@virtusa.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 for popular video game journalism websites like GameSpot, IGN, and Paste Magazine, where he writes 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@techwell.com.



JOHANNA ROTHMAN, known as the “Pragmatic Manager,” provides frank advice for your tough problems. She helps leaders see problems, manage risks, and ease the way for teams to work better. Johanna has just published *Agile and Lean Program Management: Scaling Collaboration Across the Organization*, her tenth book. Johanna writes columns for AgileConnection, TechWell Insights, and <http://projectmanagement.com>, writes two blogs on her <http://jrothman.com> website, and blogs on <http://createadaptablelife.com>. Contact Johanna at jr@jrothman.com.



A principal consultant with AIM Consulting, **BRIAN WESTENDORF** has more than fifteen years of experience in software engineering with an emphasis on mobile applications, enterprise e-commerce solutions, and content distribution systems. Brian’s experience includes app development for big-box retailers and commercial enterprises, where he has integrated cutting-edge technology into applications, including check-ins with iBeacons and loyalty cards with Passbook, and enhanced customer engagement with geofencing for in-store experiences. He lives in Seattle, WA. Contact Brian at bwestendorf@aimconsulting.com.

Your Professional Responsibility for Security and Performance

Every one of us in software development needs to take an active role in making security and performance a high priority with every project.

by **Johanna Rothman** | jr@jrothman.com

Do you know that with a common scanning app on your smartphone, you can decode airplane boarding passes? That means anyone with a smartphone can learn your name, destination, and frequent flyer number. With that information, they can attempt to hack into your airline account and retrieve your saved credit card, your address, and other identification information, such as your TSA PreCheck number or Global Entry number.

Who knew that a boarding pass—used or not—could be a gateway to identity theft? But that's not the only threat to your identity. Consider the Internet of Things (IoT). You might have a programmable thermostat that you control from your smartphone. Can someone hack into your house or steal personal information from your phone? Of course. Even TV crime shows demonstrate how easy it is to hack into software embedded in Internet-connected vehicles.

We store all kinds of personal identification on our smartphones and on websites. This makes life easier for us and for providers such as the airlines or online stores. Online airline reservations started back in the 1990s; before that, travelers had to call the airline or travel agency for bookings. I'll take the current state, thank you very much.

We automate many transactions in our society. We might call this automation or artificial intelligence. Regardless of what we call it, we see personal and professional consequences to this automation.

Security and performance are no longer nonfunctional requirements. We need to think about how to integrate both into our products. We need to know that the product is secure and scales for the intended users. I bet the reason the boarding pass barcode is not encrypted is because it takes longer to scan and verify an encrypted boarding pass. If you've flown lately, you know there is no additional time for boarding.

Considering security and performance integration as an integral piece of the product might be a new concept for everyone on a team. In the past, product managers and product

owners delegated these decisions to software architects. Now, the entire team needs to consider security and performance.

One of my first jobs as a tester was to test an operating system scheduler. The scheduler decided which task ran for how long in the CPU and when to run another task. I had no idea how to test it. I read source code, wrote a bunch of little tests, and finally had ways to overload the scheduler. I could make it crash, but my testing approach didn't really verify that it worked as intended.

That experience taught me a valuable lesson: Proving the circumstances where I could create errors was helpful, but it was not sufficient. I needed to know if the system worked. In the company, we used the under-development software. You may have heard this referred to as eating your own dog food.

I had discovered a number of edge cases our developers used to tune the system. Were my tests sufficient to guarantee the scheduler worked?

I learned then that I, as a tester, could not guarantee software reliability, performance, or quality. To see how the product was supposed to work and to discover if that were different from how it did work, I had to work with the developer to write tests we could both use. He wrote code he thought worked. I developed scenarios that helped us learn how

well the code worked—not *if* the code worked, but how well the code worked.

That's the situation in performance and security now.

We can make code work pretty well. We understand how to define requirements, how to define test cases, and how to use scenario testing. But are we creative enough to define all the scenarios we need to design and test for security and performance? We are underprepared for physical security and performance. We've heard in the news about people who kept sensitive data on their personal computers or people who walked sensitive information out of an office on thumb drives. When people act in insecure ways, all the algorithms in the world can't manage security.

“Considering security and performance integration as an integral piece of the product might be a new concept for everyone on a team.”

We may not realize the scenarios we need to address for performance. Just as I was unaware of how to test a scheduler back in the 1980s, many people don't know how to develop or test for scale.

You might have a system that previously ran using a client/server architecture but is now a cloud-based application. If so, how do you design the product and tests that scale performance from three users to thirty thousand? When is it a software application problem and when is it the infrastructure that

supports the application?

Have you, as a product team, brainstormed all the ways bad actors can use your product—or the output of those products, such as a boarding pass—to bypass security? I bet the designers of the scannable code on boarding passes didn't consider—or even imagine—that bad people would try to discover home addresses or hack into airline accounts. If they had, they might have created an encrypted code, not the plain text codes present in many of today's boarding passes.

I love the capabilities I have for on-line shopping, for managing my home, and for the future I can envision. I want a refrigerator that tells me when I'm low on milk. I want driverless cars.

We have amazing opportunities and challenges. As software professionals, we need to incorporate our new appreciation of security and performance into our products.

Let's create a safe world that works to our advantage. **{end}**



The **Map** to
your **Career Success**

Agile Expert Advanced

Get to the next level in your software testing career.

ISTQB Software Tester Certification has agile, advanced and experts paths that can show you the way, giving you recognition that sets you apart.

If you are ready to take the next step in your career, choose your software testing career path now at www.astqb.org/map.

ASTQB
American Software Testing Qualifications Board, Inc.
ISTQB Certification in the U.S.

Last Chance

to Attend Mobile Dev + Test and IoT Dev + Test

Back in 2016!

2 Conferences
1 Location

NEW for 2016!

Mobile Dev + Test

A TECHWELL EVENT

Mobile Topics Include:

- Mobile Application Testing—Training (2-Day)
- Effective Mobile Automation using Appium®—Training (2-Day)
- Develop Your Mobile App Test and Quality Strategy
- Use Selenium to Test Mobile Web Apps in the Cloud
- Designing Apps for Android Devices
- Testing Web Services and the APIs behind Mobile Apps
- Swift Programming: From the Ground Up
- Android Development Introduction: A Hands-On Workshop

IoT Dev + Test

A TECHWELL EVENT

IoT Topics Include:

- The First Wave of IoT—Blood in the Water
- Test Attack Patterns for Mobile, IoT, and Embedded Software
- Prototyping Wearable Devices Using Android
- Internet of Things: From Prototype to Production
- Turning Smartphones and Smartwatches into Hotel Keys
- Wearables: Testing the Human Experience
- The Internet of Things in Action: Anki's OVERDRIVE Racing Game
- Apple Watch, Wearables, and Mobile Data—with IBM MobileFirst

April 17–22, 2016

San Diego, CA | Westin San Diego

EXCLUSIVE COUPON
Use code **MDTR** and
SAVE UP TO \$200*

Mobile-IoT-DevTest.TechWell.com

*Valid on packages over \$400. Discount is dependant on package selected.



David Dang

Years in Industry: 21

Email: david.dang@zenenergytechnologies.com

Interviewed by: **Josiah Renaudin**

Email: jrenaudin@techwell.com

“Companies that move from packaged tools into the open source world, what they typically have is what I call a technology shock.”

“The biggest thing with Selenium is, it supports so many different programming languages and so many different platforms.”

“From a technology standpoint, the automation specialist or automation engineer has to be a lot more technical. That's the first thing that a company realizes. They need to re-choose their resources to be a lot more technical.”

“Having a framework that is program-language agnostic and platform agnostic helps from an adoption standpoint for a lot of companies being able to utilize the Selenium framework.”

“We try our best to contribute to the community. That's the reality of open source. You take something, hopefully you give something back.”

“

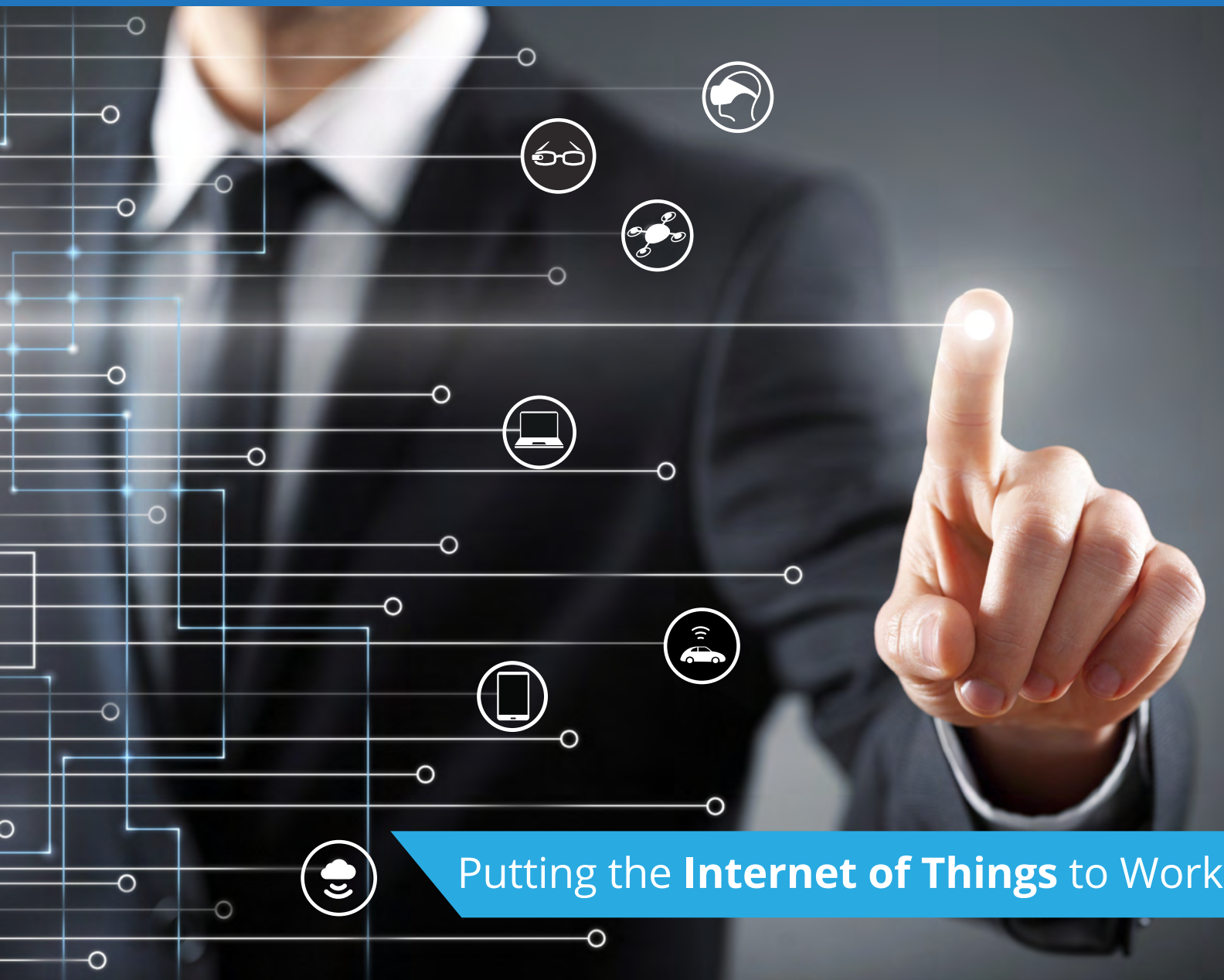
With web being bigger, with mobile being bigger, and companies transitioning into more of an agile development methodology—it's that perfect storm that really promotes Selenium and this second wave.

”

“The reason so many companies actually jump on this second wave is because it's worth it. Knowing it's worth it and knowing what it costs is really important for you to have a successful project.”

“Software development, as a whole, is moving more and more toward open source. Not just test automation, but software development as a whole. I really like the idea of having your ideas being shared and building out features that everyone can use.”

For the full interview, visit <https://well.tc/IWAE18-2>



Putting the **Internet of Things** to Work

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

Across emerging areas such as wearables, digital health and connected cars, we specialize in:

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

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

HOW TO AVOID UNNECESSARY TECHNICAL DEBT

IN

BY BRIAN WESTENDORF



MOBILE APP
DEVELOPMENT
PROJECTS



While many aspects of the mobile app market are maturing, sloppy practices can erode the foundation of mobile projects, create sustainment nightmares, and even cause projects to fail completely. In software development, the accumulation of errors in architecture, design, and poorly written code is referred to as technical debt.

Technical debt comes about when businesses prioritize short-term results over the longevity of a solution. A common adage is “You can have it better, cheaper, or faster—just pick two!” In software development, the two chosen are often cheaper and faster—to the detriment of better. Speed to market is prioritized because customer-facing features are always of more importance than cleaning up the backroom, and because development work is expensive, businesses are always trying to keep costs down. Inevitably, to make the deadline and stay on budget, something has to take a hit, and that usually ends up being the quality of the code.

Technical debt is supposed to be paid off incrementally every time new code is released, but it often continues to build because sub-standard code quality cannot be seen or felt by the application’s users, and therefore it becomes less important to the business. Imagine, for example, an electrical panel of disorganized wires held together by wadded foil and duct tape. As long as the lights turn on and the mess is hidden behind a door, what is the incentive to fix it? If that image bothers you, then you belong in the camp of people who think technical debt is a poor term that plays down the seriousness of the problem.

Without proper software design, technical debt grows at a greater rate than is optimal, resulting in increasingly higher costs to make subsequent changes to the software. The marginal cost of each software release, as well as the time it takes to complete a release, also increases, as shown in figure 1.

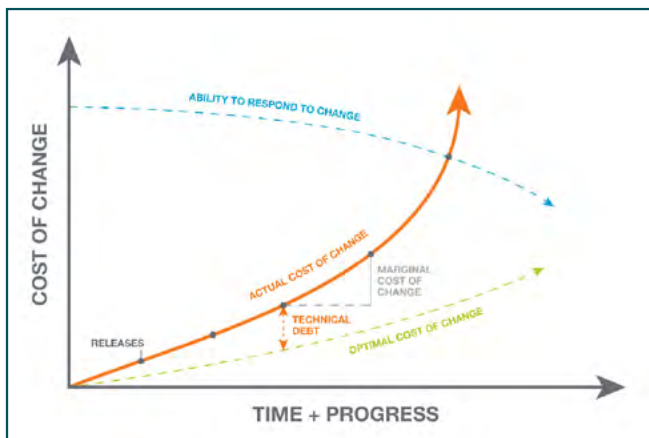


Figure 1: The impact of technical debt on software releases

If not addressed, technical debt can make an application so expensive to maintain that it takes a toll on the ability to change anything. Eventually, it will lead to total paralysis, and in order to move forward, the application must be redesigned and rebuilt at a potentially obscene expense.

Technical debt is impossible to avoid completely, but when it comes to mobile, some organizations add to it unnecessarily. The worst case is when technical debt is planted as a seed by

neglecting to lay a proper foundation before any code has been written. Although this is completely avoidable, many businesses have the perception that because mobile changes so fast, there is no need to spend time on planning. [1] When this happens, innocent errors can snowball into an avalanche.

For example, AIM Consulting was brought in by a new client to analyze why its enterprise-level iOS mobile app, under development for more than a year, had failed Apple submissions. We discovered that the code, which the company had developed in house, did not follow any discernable design patterns, was all original, and was therefore unreliable and difficult to debug. The resulting technical debt was so high that the client was actually served best in the long run by scrapping the project and starting over, ultimately costing the company millions of dollars.

This situation is more common than you might think. Too often, companies plunge into software development without properly setting up the project and end up creating something that is very costly to deploy and difficult to maintain. This is particularly true with mobile software apps because there’s a perception that the feature set is more restrictive and easier to build and deploy. Avoiding these mistakes and the technical debt associated with them becomes more manageable if, while laying the foundation for your mobile projects, you follow some basic best practices: adhering to standard design patterns, embracing open source code, and understanding the project APIs.

Adhering to Standard Design Patterns

Design patterns allow engineers to create reusable solutions. In many cases, engineers may be writing design patterns without knowing their technical definitions. Following patterns will create a solid project structure and a well-organized set of decoupled solutions. This way, when you’re in the middle of development and the project pivots to a place where you need to refactor, you know the role of each solution component. Having a solid core of design patterns and making sure your team follows those patterns helps manage technical debt.

The following are some examples of popular design patterns—not only for mobile, but for all software development:

MVC design pattern: Classes or objects in your application will take on one of three roles: model (represents your application data), view (user interface), and controller (manages the communication between the model and view by taking the data from the model and feeding it to the view for display). Conforming to this pattern helps ensure that your code is easier to understand. A common mistake is putting everything into the controller (jokingly known as the MassiveViewController in iOS). Properly breaking it up makes the code much easier to manage.

Observer design pattern: In this pattern, the observer handles and manages all communications to and between dependents. Think of a parent communicating to children either individually or all at once. Within iOS, this can be done very broadly with NSNotifications or more granularly with key value observing.

Facade design pattern: Commonly used in object-oriented programming, this design pattern provides a simplified inter-

face to a larger, more complex body of code. The façade acts as a gatekeeper of sorts, communicating only with specific components of an application at specific times, enabling efficiency in the app. It is often used in very complex or difficult-to-understand systems.

Singleton design pattern: At its core, Singleton is a very simple design pattern. It creates one instance of a class or object and gives shared access throughout the app. For example, you can imagine an app having one user and set it up using the Singleton pattern. Once established, you can access the one instance of user throughout your app.

When considering the possibility of not using a standard design pattern, it's easy to see how the technical debt in your development projects can accumulate quickly. When your project pivots, prompting discussions about refactoring, having these design patterns in place will help mitigate and organize the work that needs to be done.

Embracing Open Source Code

Another major way for organizations to avoid massive technical debt is to leverage reusable solutions from open source libraries. Open source communities have done a terrific job creating, contributing, and maintaining reusable frameworks. When developing solutions, it used to be a build versus buy discussion. Now there's typically a public GitHub or other repository where someone has already figured out the problem you are trying to solve and provided a working solution.

The important factor is knowing which to use and which not to use. When reviewing open source code, look at how often it's been updated. If the code is fresh and has been used and updated by many contributors, it's likely solid. However, if it was written two years ago and has never been updated, you are better off avoiding it. You can pull open source code into your apps, alter it in whatever ways you need, and then redeposit it to the open source collective. This makes it a better product for the community.

One point of consideration is that, while it's typical for enterprises to take advantage of reusable solutions, some organizations desire to write the code themselves for the purpose of owning the copyright to the solution or product. Creating code from scratch carries the risks of adding complexity and bugs to the app and increasing the technical debt for the project. However, depending on the project, this might be the only way to go.

The only other consideration is attribution. Most apps normally include a license view in an About menu item. Contributors usually ask that you give them credit in this section for using their open source solutions. Often the attribution is referenced by the MIT license, which is a free software license originating at the Massachusetts Institute of Technology (MIT). This license permits use of reusable code in a proprietary software application, provided all copies of the software reference the MIT license.

Understanding the Project APIs

In simplest terms, APIs allow programs to communicate with each other. In the discovery stage at the beginning of a project with a client, we need to understand what state the

client's APIs are in and how we can best leverage them. With mobile app projects, the APIs often don't exist yet. It's part of our job to help the client define the best APIs for the intended use of their data. We ensure that our clients understand the importance of APIs—how they're like the blood and nerves of an application—so we can avoid technical debt and delays later in the project due to lack of availability or usability.

The two most dominant API models for web services are Simple Object Application Protocol (SOAP) and Representational State Transfer (REST). The SOAP API was first proposed in the late 1990s and has been a popular protocol for moving structured XML data between applications. Though SOAP is still widely used today for exchanging data on the Internet, it's considered “heavy” because it requires data to be sent back and forth from application to application.

The most common API in use for mobile apps today is REST, a term often used interchangeably with RESTful. The REST API was conceptualized about the same time as SOAP and has become popular in the past several years. REST is more lightweight than SOAP in that it breaks up data so only the pieces that are needed are sent back and forth between applications. This lightweight data exchange is more responsive to user needs and requires less code—and for mobile applications, the fewer lines of code, the better.

REST has become synonymous with the term stateless. Stateless implies that no data is stored or cached when a request is made; stateful means some data is kept for use. While using an application where your next request is based on your previous request, it's better to store the data and go stateful. However, with mobile apps, stateless is the way to go because it's preferable to have less data stored locally on a mobile device. Fresh data is a prime reason people use mobile apps.

The important thing to note about APIs is you must make sure from the beginning that they are accessible and capable of supporting features and the overall project. At the start of a project, current APIs don't fully support project needs, and updates are done in parallel (or not at all). It's important to be collaborative and flexible while working toward a solution.

Avoiding Unnecessary Technical Debt

Applications with high technical debt are costly to maintain, risky to the business, and make estimations impossible. This leads to the undesirable situation where your development resources will need to spend all their time fixing problems rather than developing new features for users. You will avoid unnecessary technical debt by establishing solid practices and patterns at the start of your mobile development projects. Ultimately, when you lay a proper foundation, your development resources will be used far more efficiently, and your technical debt will be minimized and mitigated. **{end}**

bwestendorf@aimconsulting.com

**Sticky
Notes**

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

■ References

TRAIN YOUR TEAM ON YOUR TURF



BRING THE TRAINING TO YOU

Software Tester Certification—Foundation Level
Mastering Test Design
Agile Tester Certification
Agile Test Automation—ICAgile
Integrating Test with a DevOps Approach
Mobile Application Testing
And More!

60+ ON-SITE COURSES



40
TESTING
COURSES



7
MANAGEMENT
COURSES



9
REQUIREMENTS
COURSES



4
DEVELOPMENT
AND TESTING
TOOLS COURSES



17
AGILE
COURSES



2
SECURITY
COURSES

For more than twenty-five years, TechWell has helped thousands of organizations reach their goal of producing high-value and high-quality software. As part of TechWell's top-ranked lineup of expert resources for software professionals, SQE Training's On-Site training offers your team the kind of change that can only come from working one-on-one with a seasoned expert. We are the industry's best resource to help organizations meet their software testing, development, management, and requirements training needs.

With On-Site training, we handle it all—bringing the instructor and the course to you. Delivering one of our 60+ courses at your location allows you to tailor the experience to the specific needs of your organization and expand the number of people that can be trained. You and your team can focus on the most relevant material, discuss proprietary issues with complete confidentiality, and ensure everyone is on the same page when implementing new practices and processes.

IF YOU HAVE 6 OR MORE TO TRAIN, CONSIDER ON-SITE TRAINING

[SQETRaining.COM/ON-SITE](https://sqetraining.com/on-site)



Over 100+ Learning
Sessions in 1 Location!

Agile Dev Better Software DevOps **WEST**

A TECHWELL EVENT

June 5-10, 2016

Las Vegas, NV | Caesars Palace

**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

***Better Software* Magazine subscribers
register using promo code BSMCW16 by
May 6, 2016 to save up to an additional
\$400 off your conference registration**

bscwest.techwell.com

Keynotes by International Experts

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



DevOps and the Culture of High-Performing Software Organizations

Jez Humble
Humble, O'Reilly, & Associates



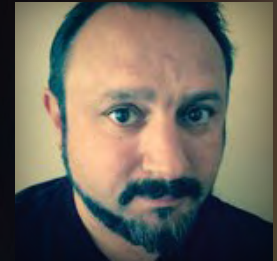
The Power of an Agile Mindset

Linda Rising
Independent Consultant



How to Do Kick-Ass Software Development

Sven Peters
Atlassian



The Internet of Things: Through the Mobile Lens

Steven Winter
Trizic

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

Configuration Management: Robust Practices for Fast Delivery

Bob Aiello, CM Best Practices Consulting

Fostering Sustained Agility

Tricia Broderick, Agile For All

A/B Testing: Improvements through Continuous Experimentation

Alan Page, Microsoft

Continuous Testing to Drive Continuous Integration and Deployment

Jared Richardson, Agile Artisans

Great Product Design with User Story Mapping

Jeff Patton, Jeff Patton & Associates

Planning to Learn and Learning from Delivery: Scrum, Kanban and Beyond

David Hussman, DevJam



JUNE 8-9

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
bscwest.techwell.com



TEST AUTOMATION

NOW WE HAVE TO GET IT RIGHT
BY PAUL GERRARD

Developers and testers have always run software to test software, but the term automated testing has been used rather lazily to denote the use of technology to execute scripted tests. Testing involves a whole lot more than just test execution [1]. In this article, I will use the term automation to mean checks [2] executed by tools, rather than by humans.

The Pressure to Automate Is Greater than Ever

Digital is the buzzword of the moment. Almost all the testing consultants I know are working on digital transformation programs. More often, it seems marketers are being handed the reins of these programs. Marketers see IT as the vehicle for doing rapid, continuous, relentless experiments in parallel with rapid, continuous, relentless delivery of new features. This is necessary to keep up with the competition.

The users of mobile applications expect them to change almost daily. The dynamics of markets that rely on mobile apps mean that consumers expect new features, offers, and opportunities. Customers conflate the notion of the novelty, dynamism, and quality of the app with the product and service they seek to buy. Users use the best apps to get what they want, and they aren't afraid to switch to another app.

In the digital domain, businesses are in an apps race. Needless to say, continuous delivery, DevOps, and pervasive automation are prominent in digital IT strategies.

Automation—Not Just of Tests—Is Critical

Clearly, digital business requires true agility. Business owners need to be confident that when the market changes, their company can respond, and when a new idea emerges, it can be market-tested in days—not months. They want to turn

ideas into working software as quickly as possible.

With continuous integration, continuous deployment, and the emergence of DevOps disciplines, developers can now promise continuous delivery. However, the fact remains that for software to be predictable, reliable, and of high quality, testers (and testing) have a critical part to play.

Developers use tools to quickly transform their code changes into deployable systems. Testers must match the velocity of transit through the deployment pipeline by providing what might be called continuous assurance by using automation in shortened timescales.

But the principal problem remains: Automation success is still hard to achieve.

Automation Ambition—and Reality

At the 1997 STARWEST software testing conference, Dorothy Graham asked attendees what tools they used. The percentages of the audience using each tool type are listed in table 1. This survey was conducted in May 1997 at the STARWEST conference in the USA and appeared in the CAST Report of 1999. [3]

The vast majority of testers had access to test execution tools, but less than one percent said they had achieved significant benefits. If the same survey were conducted today, would the results be different?

I was recently asked to comment on a study of digital IT practices. [4] The study included a question related to the coverage of business processes by automated testing. Table 2 shows that respondents mostly wanted to increase the coverage of their automated testing, usually from a low base.

In 2015 no one in the survey had achieved complete coverage, and 41 percent still had little or no coverage at all.

Category	Percent
Test execution	99
Defect tracking	80
Coverage measurement	5
Performance or load testing	30
Test input generation	3
Test management (including traceability)	20
Static or dynamic analysis	10
They were also asked how effective tools were.	
Question asked	Yes percent
Have you achieved significant benefits	<1
Do you have CAST shelfware?	45
Did you experience significant problems in test automation?	75
Do you want more CAST Tools?	75

Table 1: Survey results of preferred test tools used

Q. What percentage of your business processes are covered by automated testing and how will this change in the next 12 months?

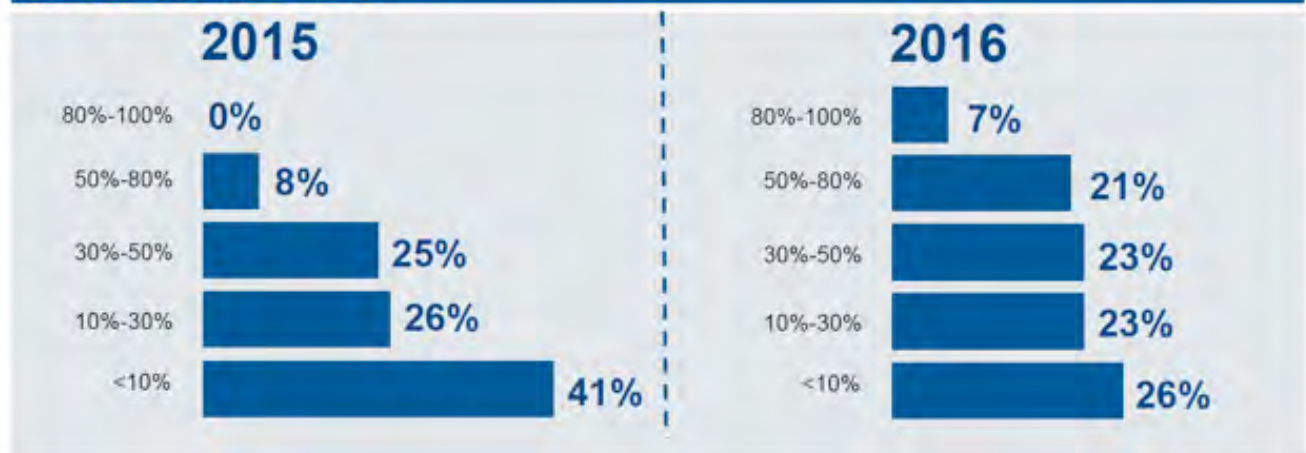


Table 2: Comparison of tester respondents wanting to increase the coverage of automated testing

Superficially, the 2016 goal seems achievable but progress during the past fifteen to twenty years has been painfully slow. There is no reason to expect progress will be any better next year.

In 2016, a lack of effective test automation may prevent organizations from achieving their digital transformation goals.

What Goes Wrong with Automation?

There are many reasons why automation goes wrong, but I'll focus on the most prominent cause of problems. Test execution tools can be programmed to detect the smallest differences in behavior, and they usually have little tolerance for error. Tools are extremely sensitive instruments and will flag unwanted changes very successfully. However, serious problems arise if the system under test is unstable. From release to release, changes in behavior made by developers are so significant that testers cannot keep pace with these changes.

Consider the use of precision instruments to measure changes by the millimeter when the object being measured moves by the meter. Imagine being the surveyor in figure 1! In that situation, a theodolite wouldn't be very useful, as you can see the movement with your own eyes. In fact, the tool will be a burden, as you try to recalibrate the tool to the object, without success. In the same way, the maintenance burden of automation often outweighs its value.

Goals and Side Effects of Automation

Usually, automation forms part of a wider anti-regression regime. [5] Although impact analysis is a proactive tactic used to minimize or prevent regression issues, automation is the more common safety net for developers to make rapid changes

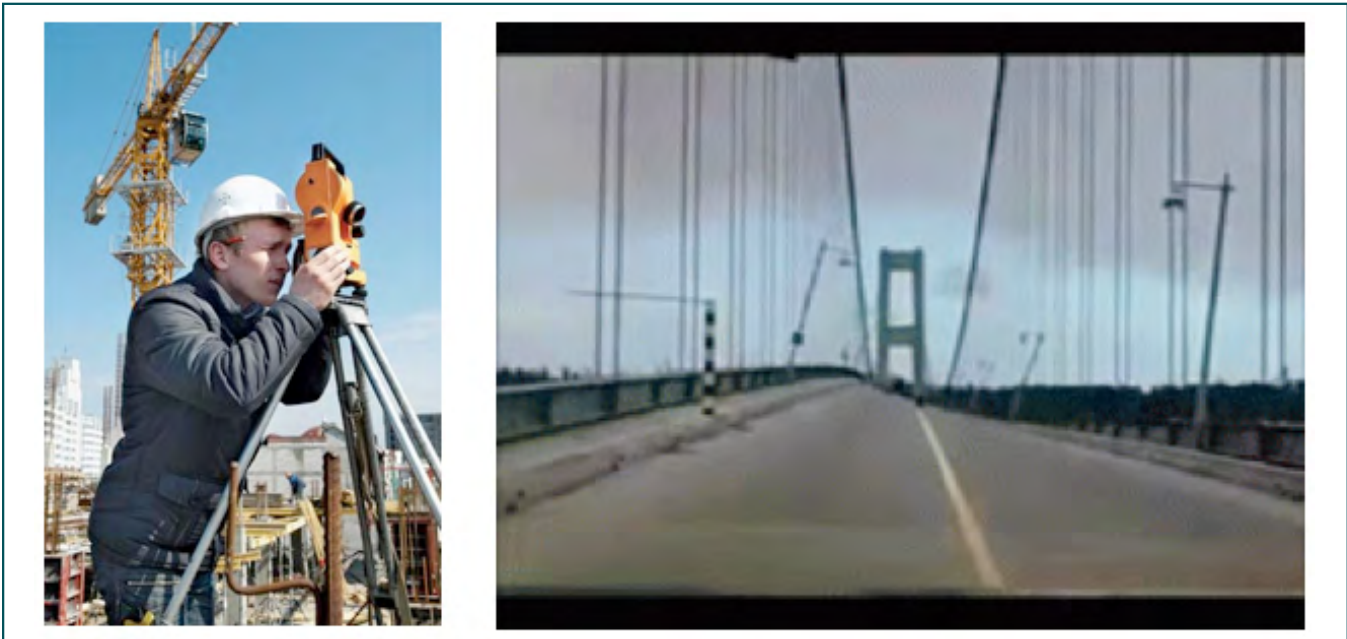


Figure 1: A laser theodolite in use; the Tacoma Narrows Bridge (aka Galloping Gertie)

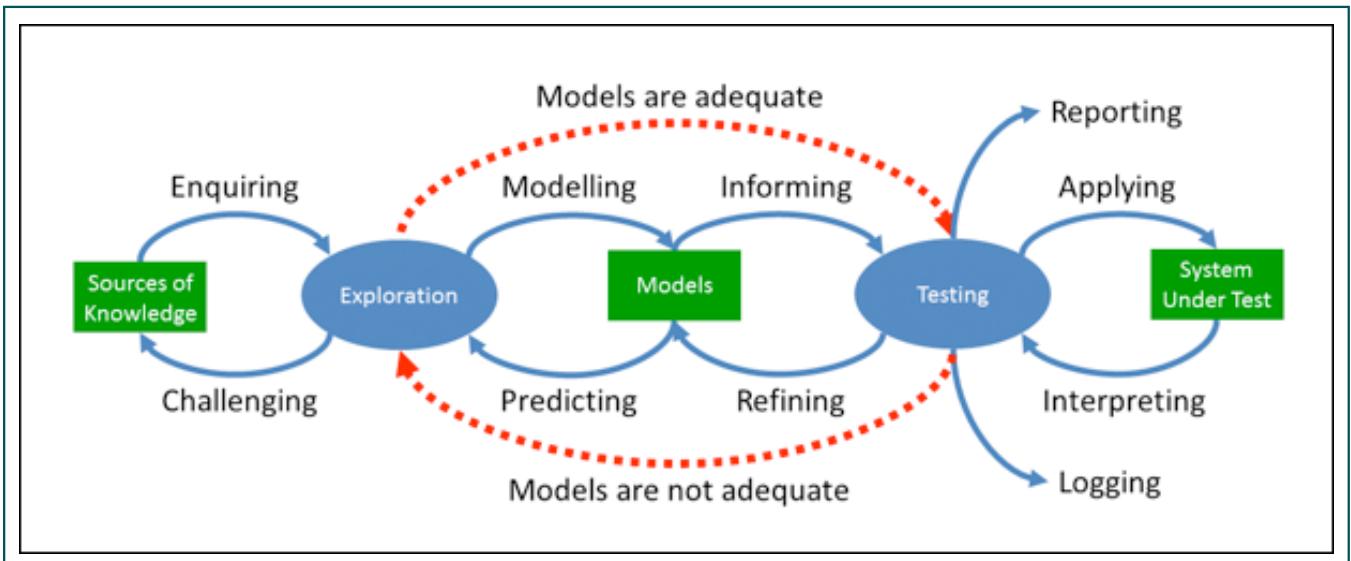


Figure 2: Ten core thought processes used in the new model for testing

with some degree of comfort.

A system and its automated regression tests are like a mold and a cast where the system and tests fit perfectly. One is derived from the other, but not without difficulty. Test-first approaches reverse the roles and simplify matters somewhat. The tests are the mold and the system is the cast. When the system changes, the tests indicate where the mold does not fit perfectly, and the system (or the tests) must then change to realign them.

New Model for Testing

Both developers and testers use their own understanding to write the software and tests respectively. I will use the term model as shorthand for this understanding.

The software and the tests are both derived from models. Developers and testers who work in silos use different models, so the software and the tests can be hard to fit. Differences and defects may be found by tests, but it is likely that a series of false alarms will be raised by the automation—possibly wasting significant developer and tester time.

To obtain a trustworthy and accurate model and to ensure the same model and understanding are shared across a team, testers, developers, and users must collaborate from the outset and continuously throughout the project lifecycle.

To better understand testing and automation, testing can be deconstructed into its constituent thought processes. In the article “New Model for Testing” [6] there are ten core thought processes of testing that apply in any context. Figure 2 has proven to be a useful visualization to explain testing concepts and processes.

In the model, the label applying represents the application (or execution) of checks by testers or tools. Test execution automation represents only a part of one of the ten processes where the thinking is supported by tools. Checks might be applied by tools, but interpretation is always a human activity.

The diagram in figure 2 reads from left to right along with parallel feedback loops. It identifies the acquisition of information from our sources of knowledge (people, documentation,

and the system under test) and understanding through test modeling. Test models might be formal and documented, but they can also be temporary, informal, mental models. These test models (like a flowchart) are used to derive examples of behavior, validate expectations and requirements (the left side of the figure), and derive checks for use by testers or an automated tool (the right side of the figure).

The left side of the model identifies the key points of collaboration between developers, testers, and users/product owners—sometimes called the three amigos. The model makes it plain where these interactions occur.

Achieving Testing and Automation Success

The goal of the left side of figure 2 is to create trustworthy models of the system, and the right side uses those models as the basis of testing.

The new model aims to make clear how the thinking on the left contributes to the thinking and activities on the right. With this in mind, we can make some simple statements that suggest how we can achieve success in testing, whether performed by humans or tools:

- The thinking on the left side contributes to the definition of features and how you test them. For example, testers can create scenarios from requirements to feed back to users as concrete examples to check that both testers and users have the same understanding of the requirement.
- Your opportunity to flush out muddled, loose, incomplete, flaky requirements is on the left side.
- The models created on the left are the blueprints for the tests and automation on the right side. The examples used to validate the requirement can be re-used as checks of the software.
- Automation won't work if the thinking to the left is not in place.
- If you aren't involved on the left, you aren't in a good position to test—with or without tools.

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.

The series of statements represent the core definition of shift left. To achieve sustainable success in testing and test automation, we need to position the thinking on the left side of the model to the left of our development processes. All who test use this thought process.

These thought processes happen regardless. Shift left simply means they occur earlier in your development process, requiring a more collaborative culture that should result in more accurate, trustworthy requirements, models, and systems.

Test Development Detects Bugs; Execution Detects Regressions

Any tester who has created automated tests for existing software quickly discovers that the process follows a certain pattern. There's an uncertain period where the function of the software is unclear, but through exploration, a clearer understanding of the software and its purpose unfolds. Scripts usually require several attempts to get them working. Clarifications are sought and provided. The tester encounters bugs, some of which are fixed. The scripts eventually align with the software (including the bugs), execute without failure, and are merged into a regression suite.

Most bugs tend to be found during this script development phase—not when the scripts become part of the regression suite. So the value occurs early, during the exploration and model-making activities.

This phenomenon is well known, and I clearly recall its being debated at EuroSTAR 1998. The new model simply makes it easy to recognize why it occurs. More importantly, we can see why teams use test-driven and behavior-driven approaches. In fact, these teams use modelling to enhance their understanding and create automated checks before any code is written.

Summary

Digital transformation is having a profound effect on how business is done and the systems required to deliver enhanced customer experiences. The demand for flexible and rapid delivery of software and systems means that test-driven, continuous delivery, DevOps, and experimental approaches are being adopted. But these approaches depend on developers and testers making more effective use of tools for test execution. So far, the industry's track record in test automation is not good enough.

In order to succeed, a new way of thinking about testing is required, and the new model of testing offers a way of identifying the elements of the test process that must be shifted left. This does not necessarily mean testers move left; rather, it is the testing thought processes that must move. This is why business analysts, users, and developers are taking more responsibility for testing.

In digital projects, the role of automation in all test types is significant, so now we have to get it right. **{end}**

paul@gerrardconsulting.com

Sticky Notes

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

■ References

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



VALUE METRICS FOR AGILE GOVERNANCE

BY MIKE HARRIS

Agile implementations, particularly Scrum, are rich in simple, team-level metrics such as story points, velocity, and burn-down charts. Unfortunately, these team-level metrics are not very useful for planning or monitoring across an entire software development organization. There is often a gap when attempting to measure an organization's efficiency, economy, and effectiveness.

Software value visibility metrics are a better choice for governing your agile software development organization-wide. In this article, you will learn how to create top-down agile metrics that are compatible with and extend traditional waterfall metrics.

Current Practices

The following metric definitions are examples from a real waterfall organization:

Delivered as promised: Defined as the software delivered that must have the functionality promised in the requirements section of the project scope agreement.

Productivity: Defined as the number of function points delivered divided by the total number of work years of effort charged against the project from start to completion.

Timeliness: Defined as a comparison of the agreed-upon baselined implementation date against the current implementation date of record.

Quality: Defined as the number of defects delivered into production.

Accuracy of effort estimate: Defined as a comparison of the originally agreed-upon baselined effort estimate against the current effort estimate.

Even though these metrics can be captured initially at the project level, they can be readily aggregated up to the program and organization level for governance purposes. As a result, project progress can be easily summarized. Without any significant understanding of software development methodologies, any executive would be able to read and understand trend lines for these metrics on a dashboard.

In comparison, let's look at how agile teams measure value. As part of sprint or release planning processes, some agile teams ask the product owner to assign relative value sizes, or value points, to individual stories or epics. From a governance perspective, these value points do not aggregate well, and few organizations even attempt to track these metrics.

The Metrics Challenge for Agile

We need agile metrics that will match the waterfall governance metrics. For example, anyone familiar with the Scrum methodology will recognize that the conversion is not necessarily straightforward for all of the metrics just highlighted. The waterfall metrics focus on assumptions built into the waterfall approach; for example, the timeliness metric assumes that it is difficult to predict when the functionality will be complete and operational. Agile, on the other hand, focuses on how much of the functionality is operational by a fixed date.

At least for the software development community, agile of-

fers better benefits than waterfall on two main fronts: value delivery and customer satisfaction. As agile practitioners and champions, we can smugly review the list of waterfall metrics mentioned earlier and observe that none of these metrics measure business value delivery or customer satisfaction directly. It is a fair observation. But, does agile really do any better?

From my experience, agile implementations tend to fall into the trap of trying to deliver to executives the same reports that waterfall delivered. Figure 1 shows a set of the default reports from a popular agile lifecycle management tool. It is not the only popular tool on the market, and I don't mean to imply that others don't have slightly better results. My point here is that the reports listed at the executive, project, and sprint levels do not include any explicit references to value delivery or customer satisfaction.

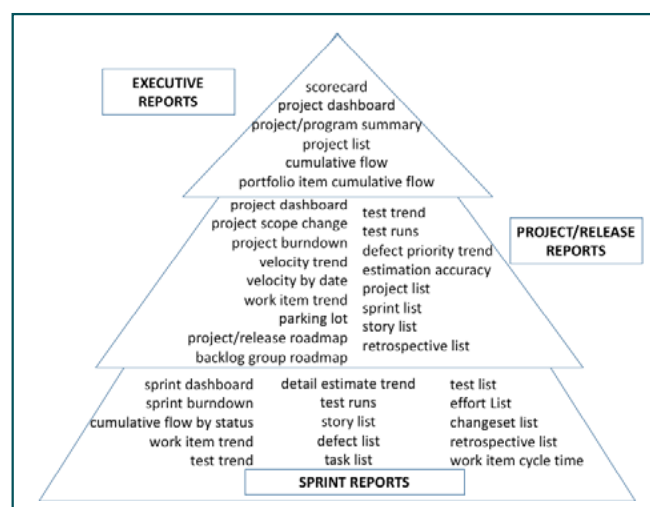


Figure 1: Default reports from a sample agile management tool

There are reports that imply business value delivery just as there is some implied value which may be business value for the waterfall metrics (e.g., timeliness). For example, at the sprint report level, work item cycle time is not a bad proxy for value delivery. On the assumption that work items are broadly similar in size, a trend to deliver them more quickly implies more value being delivered over some time period. There are two problems with this view of value delivery. First, work item cycle time is not visible in executive reports; Second, the assumption that all work items have the same business value is unlikely. Cumulative flow reports are available in both executive and sprint reporting. These reports are powerful tools for identifying bottlenecks that can be tackled to help maximize value flow. In the absence of metrics for value, they cannot explicitly report value flow.

Customer satisfaction is entirely absent from executive, project/release, and sprint reports. Maybe that's understandable in a tool that is focused on agile lifecycle management, but shouldn't it be reflected in the dashboard at the executive level? Executives need to see summaries in addition to aggregate and detailed information about customer satisfaction. To meet the needs of executives, many organizations have separate

groups and tools to conduct customer satisfaction surveys. This is better than nothing, but too often there is no connection between these high-level surveys and the agile teams who need fast, pertinent customer feedback to drive their product and process improvement.

The Solution

Organizations need to implement customer satisfaction measurements that are useful to individual agile teams and scalable for executives. This can be somewhat challenging because the best way to measure customer satisfaction is usually very situationally specific.

Let's consider how we might measure value delivery. All stakeholders and team members must know the business and economic value of the project and work toward the same goal of maximizing business value flow.

Business units and IT must collaborate to define the value for each initiative, right down to the lowest level at which resourcing decisions are made. For example, there must be an approved business case for every project or program above a certain size—let's say \$10 million in Corporation X. This business case will presumably include some metric such as return on investment (ROI), described in table 1.

ROI	Value
Less than or equal to 5%	Rejected
6% to 10%	Low, but worthwhile
11% to 20%	Medium
21% to 40%	High
Greater than 40%	Very high

Table 1: A sample project ROI

For software development to start a \$15 million project with a medium-value ROI, we need to break the project up into epics and stories. It probably does not make sense to do an individual business case for each epic and story, but we need to make sure each epic and story is linked to the master project. In addition, each epic and story should inherit the T-shirt size value label of the master project (in this case, medium).

When we think about value visualization and projects, it is important to remember that cost is not value any more than

price is. So, why do organizations only talk to IT about cost? If organizations can share business value details for their initiatives, they will enable IT to become part of the ROI solution.

Is there a comprehensive and practical way for organizations to communicate business value information for software development projects? We should first acknowledge that even simple ways to communicate and visualize business value to the tactical decision-makers are better than none at all. We are quite happy to start with T-shirt sizes, but we can do so much more with a concept known as the value visualization framework (VVF).

The Value Visualization Framework

The VVF shows a comprehensive set of steps, not all of which need to be implemented from the beginning. It also illustrates ways to implement those steps in a simple manner. The VVF shown in figure 2 is not intended to be prescriptive. Instead, it is designed to address the complaint that although assigning value to software development is worthwhile, it is just too difficult to implement in practice.



Figure 2: Value visualization framework

VVF is a five-step parallel process to the practice that creates the business case for a project and then breaks the project down into epics and stories or some form of tasks. Table 2 shows the information needed for each epic and story.

The first three steps are extremely important parts of the process and need to be well thought out. Steps four and five may need a little more explanation.

Step four—define the cost of delay—is of fundamental importance to prioritizing work packets, projects, or stories. Essentially, we should always prioritize projects with the highest

Step	Requirement
1	Define the units of value delivery (e.g., number of subscribers, hours saved in the process).
2	Define the value of the project in specific units (e.g., 17 new subscribers once deployed).
3	Define the size (e.g., 100 story points).
4	Define the cost of delay of the implementation challenge, including level of complexity, duration, and so on (e.g., \$2,000 penalty for a missed deadline).
5	Quantify the economic value once deployed (e.g., \$10, \$15, and \$20 per subscriber at weeks 9, 12, and 15, respectively).

Table 2: Requirements for each of the five steps of the value visualization framework

cost of delay. Cost of delay has three components: [1]

User/business value—relative value to the customer or business: Do they prefer this over that? What is the revenue impact, and what is the potential penalty or other negative impact?

Time criticality—how user/business value decays over time: Is there a fixed deadline, and will the customer wait for us or move to another solution? What is the current effect on customer satisfaction?

Risk reduction/opportunity enablement—what else this does for our business: Will it reduce the risk of this or future delivery? Is there value in the information we will receive? Will it enable new business opportunities?

Identifying the cost of delay for a particular story is neither intuitively obvious nor easy. There are several absolute quantitative approaches that can work when there is a strong quantitative business case or when certain facts are known. For example, a fine will be levied if the software is not operational by December 31. While it is ideal, the opportunities to get absolute monetary values tied to software development are few and far between. Instead, I recommend using a relative sizing for cost of delay.

This approach can allow cost of delay to be assigned by an informed and representative team with relatively little data. The process is similar to agile story estimation using planning poker. Usually a limited set of numbers (e.g., the Fibonacci-like series popularized by Mike Cohn [2] for use in story points: 0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100, and ?) is used by participants to select the relative cost of delay for each story against the other stories (table 3).

The example in table 3 captures the team’s judgment that story three has a higher cost of delay than the other stories being prioritized. So, all other things being equal, story three should be worked on first.

Delaying the assignment of actual market value to functionality until it is actually deployed in step five allows the team to take into account fluctuations in value due to market forces or environment changes. The example in table 4 assumes that the value of website subscribers for a website development project increases as the subscriber base grows. As a result, each subscriber is worth \$10 in week nine. This increases to \$15 in week twelve and \$20 in week fifteen. This is based on advertising revenue per subscriber growing as the number of subscribers increases. Of course, the value could just as easily fall!

The example in table 4 uses figures from a kanban simulation game. [3] Not surprisingly, the inclusion of a fine for a missed deadline affects profitability. Even without the fine, there is a real cost of delay generated because each story has a cost of delay that also impacts profits. Cost of delay should be tallied and tracked if those stories are not finished, even if the numbers are just relative, as in our example (table 3).

The granularity of these numbers can be off-putting—how can this level of accuracy be achieved in a practical implementation?

The answer is that this level of accuracy is not necessary in a practical implementation, but approximations are worthwhile and very useful if this general structure is used.

Story	Cost of Delay
Story 1	5
Story 2	1
Story 3	8
Story 4	5

Table 3: Relative cost of delay using a modified Fibonacci set of numbers

Agile Governance Metrics Maturity Evolution

The biggest objection to measuring value delivered by software development is that it is just too difficult. I am often told that software development teams can’t map the details from business cases to their work once details are broken down. This issue becomes even more cumbersome if business cases are not shared with development (or there are no business cases at all). The VVF shows that using value for decision-making can be relatively simple, and the business can be involved in a way that’s simple for them, too. That said, there is no need to attempt all of this at once. Agile governance metrics can and should evolve over time. Figure 3 provides an evolutionary path to full agile governance through the three levels of organizational metrics maturity.

The relative positions and level boundaries of the metrics in figure 3 are certainly debatable and may be different depending on the organization.

Also, some of the metrics, such as velocity and size, are fundamental to team performance and do not lend themselves to being aggregated. They are usually team-specific.

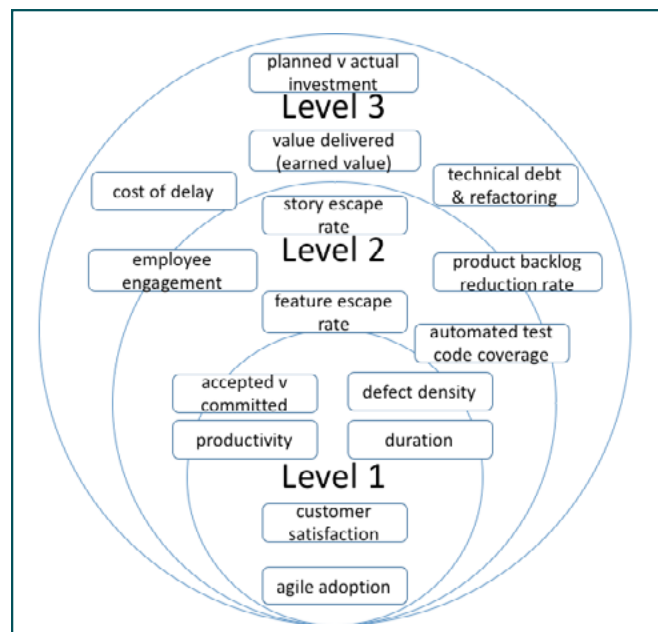


Figure 3: Agile governance metrics maturity evolution

Stories	Week 9	Week 12	Week 15
New subscribers this billing cycle	23	48	35
Total subscribers to date	23	71 (=23+48)	106 (=71+35)
Billing cycle revenue	\$230 (=\$10 * 23)	\$1,065 (=\$15 * 71)	\$2,120 (=\$20 * 106)
Fines or payments	-	-	-\$2,500 (fine for missed regulatory deadline)
Billing cycle gross profit	\$230	\$1,065	-\$380 (=\$2,120-\$2,500)
Cost of delay refund	-	-	-\$400 (cost of lost subscribers due to unduly delayed stories)
Total gross profit to date	\$230	\$1,298 (=\$230+\$1,065)	\$518 (=\$1,298-\$380-\$400)

Table 4: Example of quantifying the economic value of stories once deployed

These metrics represent a comprehensive set of measurements for agile governance. The levels are intended to support the increasing maturity of the agile teams, so level one metrics, such as agile adoption (percent of projects using agile) and customer satisfaction, can be measured and used for governance in the early stages of agile project implementations.

Value delivered is certainly a later-stage metric, but it is no less important for agile implementations and agile governance.

Conclusion

Some of these agile governance metrics are consistent with previous waterfall metrics, some are largely redundant, and others are more powerful than those typically used with waterfall. As agile software development needs to quickly deliver more business value, so must agile governance metrics include measurement of value. **{end}**

m.harris@softwarevalue.com

Sticky Notes

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

■ References

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

SOFTWARE TESTER CERTIFICATION



Professional certifications are a tangible way to set yourself apart. SQE Training offers accredited training for the most recognized software testing certification in the industry—ISTQB® International Software Testing Qualifications Board.

Foundation Level Certification

An international survey of test engineers and managers* reports that the majority of Test Managers believe that participating in the ISTQB® certification scheme will positively impact testing quality in their projects and will enable them to provide a positive career path to their employees. SQE Training's accredited **Software Tester Certification—Foundation Level** course goes above and beyond the ISTQB® syllabus, giving you practical knowledge you can apply now.

Advanced Level Certification

ISTQB® Advanced Level qualification is aimed at professionals—testers, test analysts, engineers, consultants, test managers, user acceptance testers, and software developers—who have achieved an advanced point in their careers. No matter which advanced path you are following—Test Manager, Test Analyst, or Technical Test Analyst—you can trust SQE Training's years of experience to extend your knowledge and help you prepare for the board exams.

*<http://www.istqb.org/references/surveys/istqb-effectiveness-survey.html>



- **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



2016 SCHEDULE

Orlando, FL
May 1-3, 2016

Austin, TX
May 16-20, 2016
(Foundation + Advanced)

Louisville, KY
May 24-26, 2016

Las Vegas, NV
June 5-7, 2016

Chicago, IL
June 13-15, 2016

San Jose, CA
July 12-14, 2016

Dallas, TX
August 22-24, 2016

Washington, DC
September 19-21, 2016

Anaheim, CA
October 2-4, 2016

Tampa, FL
October 17-19, 2016

Chicago, IL
October 24-28, 2016
(Foundation + Advanced)

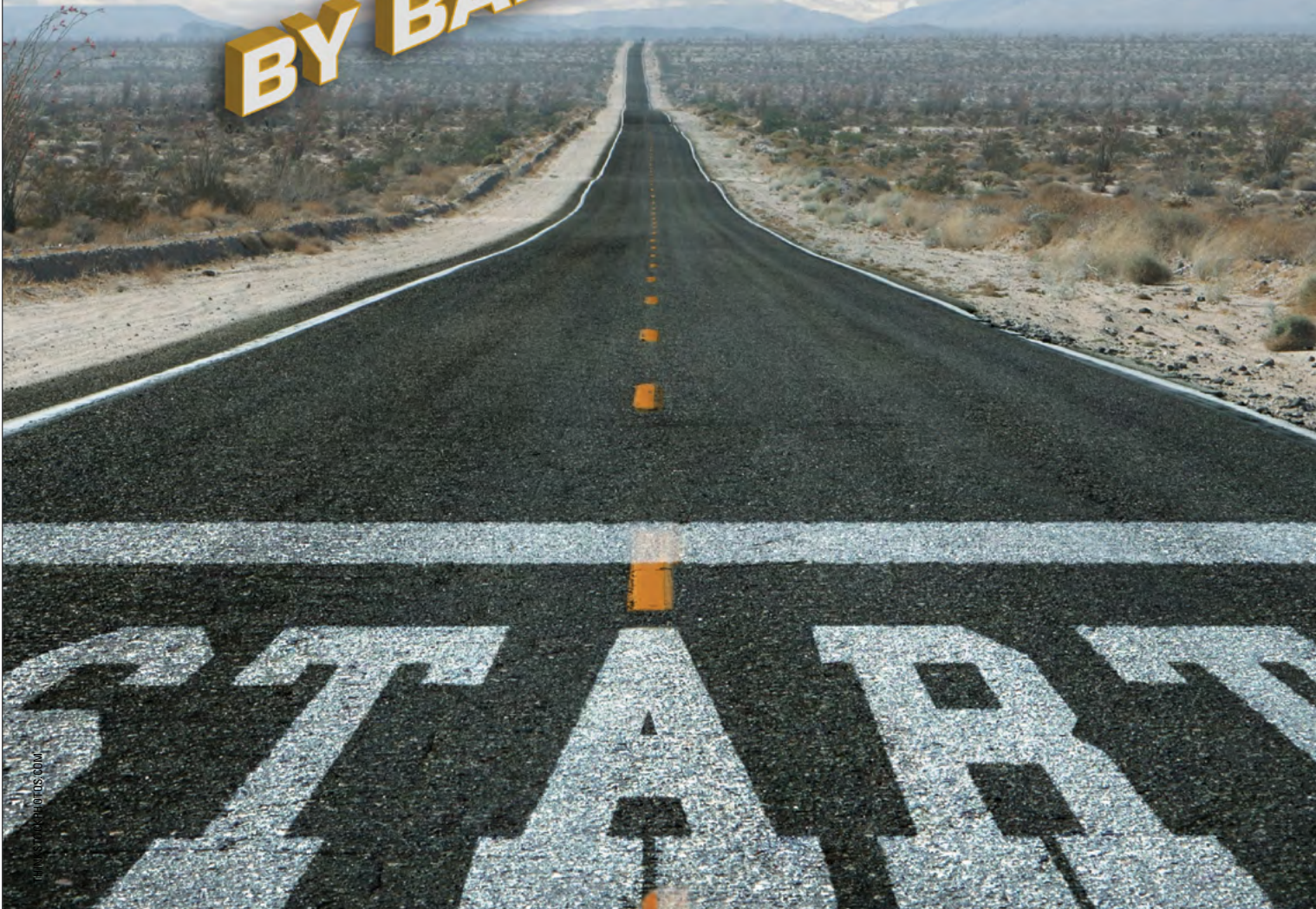
San Francisco, CA
November 7-9, 2016

SQETRAINING.COM/CERTIFICATION

 **SQE TRAINING**
A TECHWELL COMPANY

THE BENEFITS OF EARLY PERFORMANCE TESTING

BY BALJEET BILKHU



© 2016 TechWell.com

The past decade has seen a dramatic rise in how technology is being used in education, making it easier for thousands of individuals to access content that used to be scarcely available. With the advent of learning management systems in higher education and other training programs, instructors are able to collaborate and interact with students more efficiently and productively.

This ability to reach out to thousands of users comes with an inherent requirement that these systems provide a reasonable user experience. Part of this experience is ensuring that the system's functionality is correctly validated through rigorous software testing. Even though software has been tested and confirmed to be functionally correct, this may not hold true when thousands of users are accessing it at the same time. This is where performance testing comes in.

Incorporating Early Performance Testing

During performance testing cycles, some key metrics are measured, including response times and utilization of key resources such as CPU, memory, disk, and network. These measurements show whether software products that are about to be released are reliable and meet or exceed customer requirements.

A software development organization will typically have its own dedicated team to facilitate performance testing on its software products. Once the software has been developed, these teams are expected to run a full suite of tests to the point that at least 90 percent of the functional testing has been completed.

Unfortunately, leaving performance testing until late in development is risky. If a high-priority defect is discovered through performance testing, it could require a fundamental change to the architecture of the software under test.

Employing performance testing early, as features are being developed, can reduce or eliminate drastic changes to the architecture. Executing a small set of performance tests on functionality as it is being developed also will isolate changes and provide timely feedback to the engineering team.

So, realistically, when is a good time to start performance testing?

Working collaboratively with the development team and knowing when certain features are intended to be implemented will help determine this point for each team. Surprisingly, executing performance tests too early may not take advantage of key features that could address performance concerns. For agile development teams, keeping the lines of communication open is essential to ensuring that performance testing is performed at the correct moment.

For some time now, we've incorporated early performance testing in projects involving various microservices that were being developed. The results from this early testing uncovered performance issues that we were able to monitor during the course of the project and eventually verify a drastic improvement when the development finished.

We created our performance tests once the APIs were established for the microservices that were being developed for the project. Seeing that these microservices would be used by thousands of users, we ensured from the beginning that performance

was a high priority. Initially, we used automated tests (written using Apache JMeter) to ensure these APIs were functioning correctly. After more functionality was implemented, we employed the same tests in a performance manner by adjusting the number of threads (users) and loop counts for these tests.

Because the microservices being developed were outside the main learning management system (LMS) software, we were able to see how these APIs performed as a standalone entity without any interference or dependency with the LMS. After developing and executing these API tests, we were able to develop a set of performance tests that would simulate end-user actions, specifically focusing on the actions that a particular user set viewed as critical to their day-to-day operations.

At this point, the testing team still did not have the final user interfaces to verify, but we were able to employ our performance tests on existing legacy views and controls that would be used, and exercise the underlying, newly developed code. This is an example where early testing offers great benefit to the team.

Test Procedures That Validate Performance

The test procedure we employed in executing our performance tests included the following procedures:

- Get training on the performance tools currently being used by the performance team
- Develop a baseline performance characteristic based on current functionality
- Execute the same performance test suite with the newly developed features enabled to ensure compatibility
- Continue executing this test suite with increased load to find out the inflection point where response time becomes unacceptable
- Execute the test suite on a regular basis (e.g., at the beginning of each sprint) to make sure that functionality doesn't break

For this project to be a success, the performance team agreed to incorporate earlier performance testing within the development cycle. Not surprisingly, the performance team in our organization was thrilled to help us in this endeavor. From providing the background theory on performance testing, helping to provide a tool and training, reviewing our initial results, and answering our many questions along the way, the performance team was actively involved and essential. Afterward, the performance scripts that were developed became regression tests for this team, allowing us to continue developing new tests for upcoming functionality.

When establishing a baseline, we took a snapshot based on functionality that we will continue to use when the new features (like "adding content to a course") are incorporated. After developing the scripts that mimicked this action, we selected an arbitrary number for the loop count (number of times this test would be run) and set up the system to monitor key statistics (CPU and memory utilization). This test was run several times to ensure the results were consistent.

Having established a solid baseline, the same performance test suite was then executed to identify any performance impacts

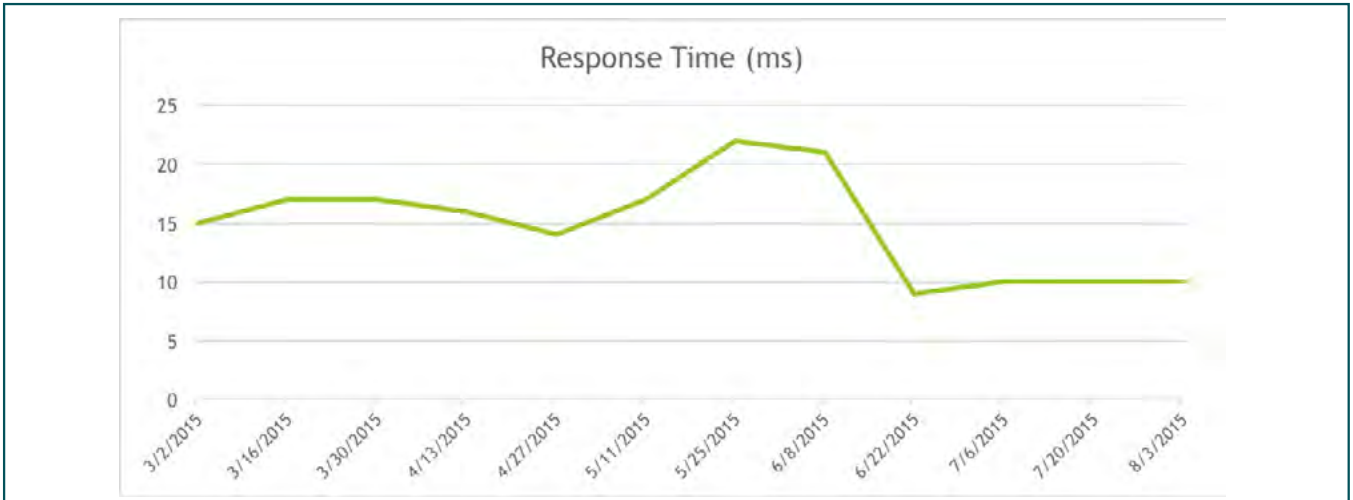


Figure 1: A microservice's performance monitored across the entire project

when additional functionality was implemented. To our surprise, we noticed a significant degradation in the performance on a particular action that the test was performing. The development team was notified and was excited that this issue was found so early in the development cycle.

Scheduling When to Run Performance Tests

At the start of a project, a schedule is established to rerun these performance tests at the beginning of every two-week sprint. The tests verify whether any changes in the previous sprint caused any change in the overall system performance. With such a schedule in place, tests are executed continuously and the team analyzes results of the system performance. Establishing this schedule ensures that any reported issues are properly addressed and that performance falls within expectations with the changes being introduced. As shown in figure 1, the performance of a particular microservice was determined over the course of the project. The performance was identified as degraded due to some unexpected code changes, and the issue was escalated to the project team. As a result, the team was able to address the problem and correct the performance issue.

With the introduction of new functionality, we're always on the lookout to quantify what would be an unreasonable response time for end-users. By rerunning our response time tests, we can identify values where degradation impacts overall system performance. Having this data allows us to present our results more intelligently and objectively to our product management and development teams. This helps ensure that product decisions are made to address these concerns.

While I haven't seen a standard set of requirements for web application response guidelines, Jakob Nielsen provides some advice that can be used in performance testing. [1]

- 0 seconds: users need to experience an instantaneous feeling regarding their actions
- 1 second: upper limit for a user's train of uninterrupted thought
- 10 seconds: upper limit for keeping a user's attention focused responding on a dialog box

Detecting and Fixing Performance Issues

Performance testing can be successfully performed on a well-established baseline once a performance test suite with acceptable response times has been defined. We choose to run these tests at the beginning of each sprint. When issues with performance are uncovered, a user story or defect should be created to address them earlier rather than later. The benefit of this approach is twofold:

1. Capturing the system's performance data allows us to view the behavior once certain user stories are implemented by isolating the possible root causes to the items delivered within the past sprint.
2. Resolving any performance issues immediately reduces the risk of inherent performance issues becoming more difficult to fix as the project continues to be developed.

Moving Forward

Using this fast-paced, iterative approach to test performance, we experienced improved overall system performance as we placed more emphasis on testing earlier. Showing the testing results to the engineers has helped bring them on board in this initiative. And as a result, collaboration couldn't be better. We also noticed that more emphasis was being put on developing infrastructure (under-the-hood) user stories that take into account performance aspects in the design and implementation phases. This has led to a more proactive approach in building quality into our products with a performance mindset from the beginning. {end}

baljeet_bilkhu@hotmail.com

Sticky Notes

Click here to read more at StickyMinds.com.

■ References

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!

How DevOps Is Making Testers Evolve

By Adam Auerbach

The DevOps movement focuses on bringing together development, testing, and operations so that feature teams have full accountability from code to deploy, with the dream of continuous delivery. But with this revolution, more and more companies are starting to abandon “traditional testers” and bring on software developers for testing.

Does this mean a tester will no longer be a role on the team? No—but it does mean testers have to evolve and ensure that quality is baked in.

The first part of the evolution is moving from being a good tester to a good team member. Get on the same page as your product owner and developer, and ensure you are aligned on scenarios prior to coding and testing. (Utilizing behavior-driven development or acceptance test-driven development is a great way to do this efficiently.)

Keep reading: <http://well.tc/3gmm>

The Potential Problem with Hiring Smart People

By Naomi Karten

I'd have thought being smart is a prerequisite for most jobs. OK, maybe not jobs filled with routine, repetition, or mindless activity. But intelligence certainly seems to be a minimum requirement for jobs that require careful analysis, critical thinking, problem-solving, or decision-making.

But apparently, intelligence is not always an asset. For example, intelligence can breed hubris, excessive self-confidence, and a superiority complex. These qualities can lead intelligent people to assume they're right when they're not, and to be so certain that their decisions are correct that they reject anyone else's perspective.

Keep reading: <http://well.tc/3gms>

The Internet of Things Moves Software Testing into the Physical World

By Jon Hagar

More than IT, PC, or web testing, testers working with IoT devices need to understand the physical world of testing as well as the logical world of software. So, what is the physical world? How is it different from the logical world we're used to, and how will devices interface with the logical world?

First, some history. Mankind has been using tools since the Stone Age. Our physical tools have become more sophisticated in the millions of years since, but it has only been in the last sixty years or so that we have started merging the physical and logical worlds.

Keep reading: <http://well.tc/3gme>

Agile Does Not Equal Scrum: Know the Difference

By Johanna Rothman

I have a problem with the way many people talk about Scrum. They say “agile/Scrum,” as though they're the same thing.

No, no, no. Scrum is just one way to approach agile. It's a terrific project management framework, especially for cross-functional teams that work on one project at a time.

Scrum does not say anything about the technical practices. The team is supposed to retrospect to determine which practices they need to add, subtract, or change. The team is also supposed to remove its impediments as it proceeds.

Keep reading: <http://well.tc/3gmn>

Testers, Tech It Up! Become More Technically Competent

By Michael Sowers

If change is one thing that is certain in life, then a continuous focus on learning, growing, and honing the skills of the testing role is an absolute.

In my early days as a tester, I had the option of becoming more technical or becoming more of a domain expert. On some projects I'd choose a more technical role, and on others I'd strive to learn all that I could about the business process and contribute as a subject matter expert. The technology you evaluate, of course, dictates your flexibility in this regard. The challenge of having one foot on the technical side and the other on the domain side is that you may only develop average competency in both.

Keep reading: <http://well.tc/3gmh>

Accelerating Your DevOps Processes with Agile

By Bob Aiello

One theme we consistently hear is the urgent need to accelerate the application development process. Agile has done a lot to help meet this goal, in part through effective iterative development.

In teaching configuration management and DevOps continuous delivery, I often observe that while everyone is striving to be agile, few people can actually verbalize the guidance described in the Agile Manifesto—and even fewer can really discuss agile principles. Going faster without understanding whether we are going in the right direction has a significant risk of our getting lost. To really achieve development velocity, we need to understand agile in a pragmatic and hands-on way.

Keep reading: <http://well.tc/3gm7>

Why User Interface Test Automation Is Worth Doing Well

By Jason Hagglund

Let's be honest: User interface test automation isn't always the most glamorous work. It tends to be repetitive. No end-user of your product will ever interact with test automation code. It can also be hard to produce reliable, stable UI tests compared to other kinds of automated tests. Given these conditions, it is understandable why UI test automation efforts sometimes fail—either by falling into disrepair due to a lack of maintenance, or by not being undertaken at all.

I propose that in spite of the above, UI test automation is worth doing, and worth doing well—with the same high standards to which we would hold our application code.

Keep reading: <http://well.tc/3s2s>

Getting Started with Wearable Devices? Consider This Platform

By Lance Gleason

Software development has undergone a series of major technological transformations during the past three decades. We saw the desktop revolution in the late '80s, followed by the Internet revolution in the '90s, and finally the mobile revolution, which started in the mid-2000s and continues to evolve.

Of late, we're also seeing a new type of revolution, aptly referred to as the Internet of Things. This term is, essentially, a broad description for a new commodity class of connected sensors and devices. One particularly fast-growing segment of the Internet of Things is wearable devices. At CES 2016, there were multiple vendors who demonstrated and introduced the latest innovations in fitness trackers, smart watches, and even connected apparel.

Keep reading: <http://well.tc/3sxc>

Use Process Improvement Methods to Elevate Your Test/QA Workflow

By Claire Lohr

Many quality assurance testers face the same challenge: an ever-increasing number of functions, interfacing systems, and environments to support (without the old environments going away much). Unfortunately, this does not seem to lead to more time to do the job—in fact, they often have less time, with more last-minute changes.

What can help? Automation is great, once it works; the first time takes approximately three times as long as doing the testing manually. The benefit is forever, but the automation tools and systems still need maintenance. Adding testers has diminishing benefits, as the cost-estimating models show us (SLIM-Estimate or the COCOMO II Cost Model). The overhead of communicating with all testers reduces the benefit of each added tester as the team increases in size.

Keep reading: <http://well.tc/3s2h>

What's a Tech Lead? Decoding This Developer Role

By Steve Berczuk

The role of technical lead can be hard to define, and in many cases people accept the role without knowing what the definition is. Patrick Kua proposes one definition, but not all organizations share the same interpretation, or even define the role explicitly.

Essentially, a tech lead is someone with a programming background who leads the development team. The tech lead deals with technical matters within the team, works with stakeholders to define requirements, and communicates with the other developers to decide on a project vision.

Keep reading: <http://well.tc/3s2e>

Where Do Software Bugs Come From?

By Matthew Heusser

I was discussing the new program to guarantee bug-free software with a friend, an agile coach I respect. He pointed out that there are basically two places bugs come from: either a programmer screwed up, or there are missing, misunderstood, or misinterpreted requirements. According to my friend, the first group can be driven to near zero with modern techniques, while the second is “always ten years away.”

Keep in mind, this is some who takes software development seriously, as craft.

It makes me wonder if he's ever opened up a bug tracker, looked at the last hundred bugs, and tried to categorize them. I have, and I've found a few more categories than just those two.

Keep reading: <http://well.tc/3s2n>

Five Misconceptions about Test Automation

By Hans Buwalda

Through the years, I have heard many valuable opinions about test automation, and I want to give my own twist on some of these. Let me start by providing my view on automated testing. I have pioneered keyword-driven testing, and its culmination led to a method called action-based testing, which is supported by—but not dependent on—a product called TestArchitect.

In action-based testing, the focus is on a modular approach, where tests are organized in test modules, each with a clear and differentiated scope. The test cases in the module consist of sequences of “action lines,” each starting with an action keyword (short “action”) with optionally one or more arguments. In action-based testing, the automation efforts focus on automating the actions rather than the tests. What I've learned is that using actions shifts the focus from technology and instead places it on test design as the key driver of automation success. In other words, a clever automation engineer alone is not enough to fix a situation where tests are not well designed.

Keep reading: <http://well.tc/3e3r>

ZAPTEST

Any Software Cross-Platform

Automate testing of ANY software user interface AS-IS!

ZAPTEST can automate any type of GUI based software application with no limitation Mobile, Web, Desktop

Browsers, Mobile native apps, POS, SAP, PDF, Oracle, POS, Liquid UI, Windows, iOS, Android, Windows, WPF, WinForms, Chrome, Safari, Firefox, IE, Chromium, Opera, Flash, Java, SVG, XML, Mac OS X, BlackBerry, Terminal, Shell, Qt, MFC, GTK+, Cocoa, .NET, OpenGL, Flex, jQuery, AIR...

1 Test Script = 1 Business Process

Reduce number of test scripts for test development and maintenance

Auto-generate Test Documentation with one click from automated scr

No need for development manual test cases

Execute 1Script across ALL testing environments Cross-Platform at once in PARALLEL!

Same test script runs on all testing environments: Mobile and Workstations

Expedite time to Regression testing

Support test process for AGILE / DevOps!

Start Shift-Left testing at the SDLC Design phase

Use application mock-ups for test development

Start test cycle with test automation

Integrate with test management technologies: HP ALM & CA Rally!

Keep test management within your corporate standard

Convert automated scripts into manual test steps

Execute test automation and integrate test results for reporting



GET STARTED!

Embracing the Top Trends in Software Testing

As mobile, cloud, and IoT technologies are taking over, traditional software testing approaches may no longer apply.

by **Shyam Ramanathan** | sramanathan@virtusa.com

The proportion of IT spend allocated to QA and testing is predicted to rise to 40 percent by 2018 according to the World Quality Report for 2015–2016. [1] On average, organizations spend a lesser portion of their IT budgets on testing, but avoidable costs of failure often account for more of their IT budgets. These crippling expenses make it imperative that organizations stay ahead of key software testing trends to save money and prevent failures.

Consequently, leading software development companies are adopting these eleven proven practices not only to avoid unnecessary costs but also to improve quality.

1. **Establish a true testing cost of excellence (TCoE):** QA and testing budgets, as part of the overall IT budgets, have leapt to 33 percent, an 11 percent increase from 2014. In. TCoEs allow organizations to consolidate their testing operations in response to market demands and differing models of testing delivery. The keys to building a successful TCoE are getting sponsor buy-in; evangelizing with project teams; defining end-to-end communication channels; and establishing clear processes, templates, reports, and metrics.
2. **Embrace mobile testing:** Mobile devices are here to stay. Today’s main focus in mobile testing is managing various devices and operating systems. Any organization that can establish a mobile testing lab, take ownership by managing devices, and automate test coverage on mobile devices will have an edge in the current business landscape.
3. **Automate your tests:** Automated testing has been around for a long time and will continue to thrive due to its ability to reduce user error during the testing process. Test automation can also help the customer develop trust in your software product. There is a shift away from licensed software automation tools toward open source tools like Selenium, which gives organiza-


“Finding expertise in the specialized area of security testing is critical to establishing a well-rounded testing organization.”

tions with expertise in open source tools an edge in the emerging market. It is imperative to do a proof of concept to ensure that the selected tool works successfully in the client environment.

4. **Establish metrics:** There are different schools of thought with respect to metrics. Some say software testing is a creative endeavor and that it is not useful to measure creativity. But all stakeholders want to know where their projects stand, and establishing metrics to measure progress is critical to the overall success of the project. Metrics can be used to measure both product quality and the testing itself. Without metrics, how do you know when your quality goals have been achieved? Reporting these metrics on a regular basis ensures project quality is always maintained, and the stakeholders are well informed. This is vital to helping the customer make project decisions and appropriate course corrections, which will eventually lead to better business outcomes.
5. **Invest in security testing:** Given the publicized security breaches with Target and other big-name companies, security has come to the forefront of testing. Cloud-based testing has recently slowed due to security concerns. This is because public clouds share resources between different organizations, and virtualization creates a lot of vulnerabilities. The best way to overcome this is to ensure adequate controls are in place to secure your environment. Finding expertise in the specialized area of security testing is critical to establishing a well-rounded testing organization.
6. **Provide an independent view of quality:** To conform to modern-day quality compliance requirements, it is becoming more common for software organizations to have separate development and QA functions, each with its own resources and procedures. Separate organizations can ensure that quality is built in and that there is no engineering bias in test results.

7. **Shift left:** IT organizations are now using a larger portion of their budget for QA. In order to ensure that the budget increase provides value and is effective, the QA team must be involved from the very beginning of the development lifecycle. The earlier defects are detected, the lower the cost of fixing them. Getting testing resources involved as early as the requirements definitions phase reduces any misunderstanding of the work to be performed, undertaken, and completed.
8. **Set up test environments early:** QA teams have to become better informed at the beginning of a project to clearly identify the environments to be tested. It may be costly, but it is imperative to have separate environments for different testing efforts, like unit testing, system testing, user acceptance testing, performance testing, and automation testing. Without separate environments, a project timeline can be unnecessarily delayed as equipment is being repurposed for testing. This wasted time can also introduce human error where systems are not properly initialized for each round of tests.
9. **Scale testing for the Internet of Things (IoT):** IoT has huge market potential that is expected to connect more than a billion devices within the next few years. A robust test strategy is needed because of the complexity of a slew of devices, regulations, and various modes of communication. Another key to enabling IoT testing is to have a reliable and secure network. To succeed in developing for IoT, we need standards and practices in place early in the adoption. Establish clear standards for various devices and sensors, clearly defining the data needs, creating a real-time dashboard showcasing various activities, and, finally, putting in place robust controls to ensure the security of all systems involved.
10. **Adapt tests to new technology:** New technologies like cloud and mobile testing are on the rise. According to the 2013–2014 World Quality Report, mobile testing has seen a rapid rise from 31 percent in 2012 to 55 percent in 2013. [2] Currently almost 36 percent of software is hosted in the cloud, but a lot of businesses still don't have the necessary infrastructure for cloud testing. This might cause businesses to opt for testing as a service (TaaS) options. TaaS is an outsourcing model in which testing activities associated with an organization's business activities are performed by a service provider rather than by its own testing resources.
11. **Validate big data:** Big data is a term that describes large quantities of data generated from a business on a daily basis. This is an emerging trend worldwide, and a lot of organizations (thanks to social media) have access to data they could only dream about in the past. The main premise is that this data can be analyzed for key insights that eventually lead to better decisions and business moves that help the organization. According to some experts, big data will account for 50 percent of the total software testing budget. The essential point for software testing teams is ensuring the highest order of security when validating a large amount of data.

This is an exciting time to be in the software testing industry. Larger budgets are being allocated to the testing effort, and every testing organization should focus on these eleven trends. {end}



Click here to read more at StickyMinds.com.
 ■ References

index to advertisers

Agile Dev, Better Software & DevOps West	https://bscwest.techwell.com	16
ASTQB	https://www.astqb.org/map	8
IoT Dev + Test / Mobile Dev + Test	https://mobile-iot-devtest.techwell.com	9
Infostretch	https://www.infostretch.com/getstarted	11
Ranorex	https://www.ranorex.com	2
SmartBear	https://smartbear.com	23
SQE Training - Live Virtual	https://www.sqetraining.com/training/delivery-options/live-virtual	1
SQE Training - On Site Training	https://www.sqetraining.com/onsite	15
SQE Training - STF/ ADV Certification	https://www.sqetraining.com/certification	29
STARCANADA	https://well.tc/starcanada2016	Back cover
STARWEST	https://starwest.techwell.com	Inside front cover
ZapTest	http://www.zaptest.com	35

Display Advertising
advertisingsales@techwell.com

All Other Inquiries
info@bettersoftware.com

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>). 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.

GET INSPIRED *in* TORONTO

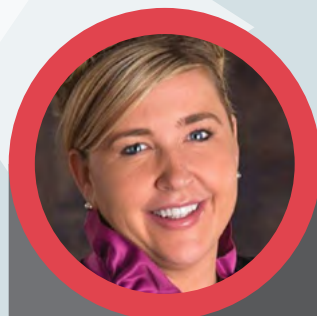
WITH THESE KEYNOTES



Testers in Agile Teams—Isolation or Collaboration?
Rob Sabourin, AmiBug.Com



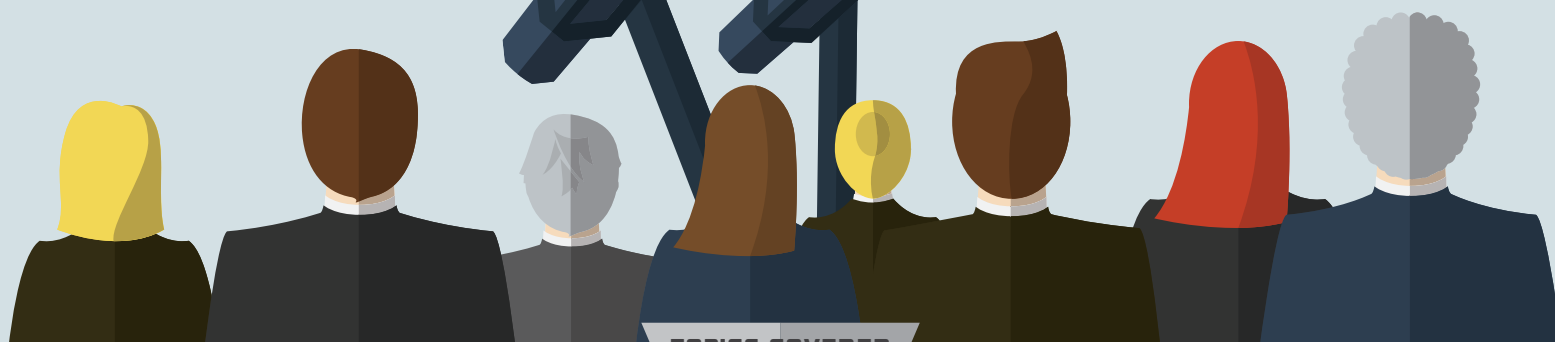
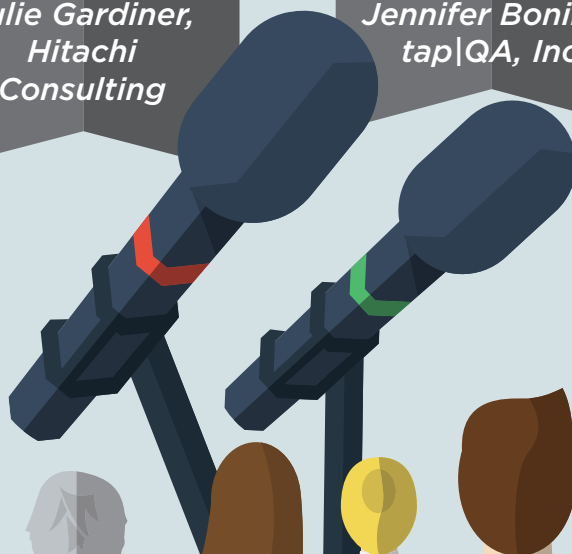
Evolution—Not Revolution: Transforming Your Testing
Julie Gardiner, Hitachi Consulting



Testing Leaders in the Digital Age: Are You Adapting?
Jennifer Bonine, tap|QA, Inc



The Future of Applications and Application Testing
Dwayne Forde, Pivotal



TOPICS COVERED

Test Management
Test Techniques
Measurement & Metrics

Test Automation
Agile Testing
DevOps & Testing

Mobile Testing
Test Data
Cloud Testing

Performance Testing
Testing the Internet of Things
People & Teams

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

STAR CANADA

A TECHWELL EVENT

October 23–27, 2016
Toronto, ON
Hyatt Regency Toronto

Learn More: <https://well.tc/starcanada2016>