

SUMMER 2017

[BETTERTM SOFTWARE]

A TECHWELLTM PUBLICATION

10

Lessons Learned in Cross-Platform Development

YOU MAY NOT BE AGILE

10 things you must do to claim the benefits of transitioning to agile

THINKING UPSIDE DOWN

Motivate your team to even greater levels of achievement



GET
INSPIRED
AT THE PREMIER EVENT
FOR
SOFTWARE
TESTING
PROFESSIONALS

STAR WEST

A TECHWELL EVENT

OCT. 1-6, 2017
ANAHEIM, CA

[CLICK HERE FOR DETAILS](#)



**TRY IT
FREE**

QMETRY
DIGITAL QUALITY PLATFORM

Accelerating Quality for Digital Enterprises Intelligently

Test Management, Test Automation, Bot Testing and Quality Analytics



Accelerate API Quality with SmartBear



SoapUI NG Pro

API functional testing tool to write, run, integrate, and automate advanced API tests



LoadUI NG Pro

Load testing tool to perform load, stress, and scalability testing for APIs



ServiceV Pro

Advanced service virtualization tool to reduce dependency on 3rd party services



Dev

API Functional, Security and Performance Testing

API and JDBC Test Environments - Service Virtualization



Ops



Scriptless



Quick



Integrated

INSIDE

Volume 19, Issue 3
SUMMER 2017

Features



16

On the Cover

10 Lessons Learned in Cross-Platform Development

Building an app for a single platform is difficult, but designing, implementing, and testing an app targeting multiple operating system platforms can be next to impossible. The secret balances upfront design with customer feedback. *by Dewey Hou*



24

The Power of Thinking Upside Down

Software developers can become bogged down trying to keep up with agile process and procedures. Get better results by rethinking your approach to balancing focus, agility, management, and testing. *by Paul E. McMahon*



30

10 Things You Must Do to Become Truly Agile

Agile is not a state of doing; it's a state of being. Adopting business models on value and learning how to make teams autonomous are both necessary steps to reap the benefit of agility. *by Jim Schiel*



38

How Technology Is Changing the Way We Learn

Modern technologies like virtual reality, cloud-based systems, and measurement of content have disrupted how we learn. Standards have evolved to improve how learning material can be published to any device. *by Troy Tolle*

Columns

09 TECHNICALLY SPEAKING

Achieving Continuous Improvement and Innovation in Software

There is tremendous pressure on software development teams to deliver software faster, better, and cheaper. Quality engineering with a focus on innovation is the answer. *by Mike Sowers*

45 CAREER DEVELOPMENT

You Get What You Tolerate

We've all worked with a talented developer who can be a frustrating challenge to manage. First-time managers may unknowingly encourage bad behavior. There are several innovative ways to resolve the situation. *by Andy Kaufman*

Departments

06 Mark Your Calendar

07 Editor's Note

08 Contributors

12 Interview with an Expert

43 TechWell Insights

47 Ad Index



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.

MARK YOUR CALENDAR



Helping organizations worldwide improve their skills, practices, and knowledge in software development and testing.

Software Tester Certification—Foundation Level

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

Aug. 7-11, 2017
Virtual Classroom

Aug. 29-31, 2017
Denver, CO

Sept. 12-14, 2017
Philadelphia, PA

Sept. 19-21, 2017
Boston, MA

Aug. 15-17, 2017
Detroit, MI

Sept. 12-14, 2017
Atlanta, GA

Sept. 12-14, 2017
Tampa, FL

Sept. 26-28, 2017
Dallas, TX

Agile Testing Training Week

<http://www.sqetraining.com/agile-week>

Sept. 12-15, 2017
Philadelphia, PA

Sept. 26-29, 2017
Dallas, TX

Sept. 19-22, 2017
Seattle, WA

Oct. 24-27, 2017
Washington, DC

Mobile Testing Training Week

<http://www.sqetraining.com/mobile-week>

Sept. 12-14, 2017
Philadelphia, PA

Nov. 14-16, 2017
San Jose, CA

Oct. 24-26, 2017
Washington, DC



Conferences

Cutting-edge concepts, practical solutions, and today's most relevant topics. TechWell brings you face to face with the best speakers, networking, and ideas.

STAR WEST

A TECHWELL EVENT

October 1-6, 2017
Anaheim, CA

[LEARN MORE](#)

STAR CANADA

A TECHWELL EVENT

October 15-20, 2017
Toronto, Canada

[LEARN MORE](#)

Agile Dev
Better Software
DevOps EAST

A TECHWELL EVENT

November 5-10, 2017
Orlando, FL

[LEARN MORE](#)

STAR EAST

A TECHWELL EVENT

April 29-May 4, 2018
Orlando, FL

[LEARN MORE](#)

Agile Dev
Better Software
DevOps WEST

A TECHWELL EVENT

June 3-8, 2018
Las Vegas, NV

[LEARN MORE](#)

Rethinking How Software Is Developed

This *Better Software* issue will inspire you to rethink how software development should be approached. Building software for multiple platforms at the same time has always been difficult. Our cover story by Dewey Hou, “10 Lessons Learned in Cross-Platform Development,” will remove the mystery of how to do it—and Dewey should know. He leads the development of some amazing Windows and macOS commercial software products.

Both *Better Software* magazine and TechWell Corporation embrace the benefits of agile software development. However, most of us have learned the hard way that “being agile” is never enough. That is why I’m so impressed with Jim Schiel’s approach to agility in “10 Things You Must Do to Become Truly Agile.” And Paul McMahon’s “The Power of Thinking Upside Down” may be the inspiration you need to motivate your team to greater levels of achievement.

The trend for online learning is augmenting, if not replacing, the traditional classroom. So how exactly do cloud-based courseware solutions work? Troy Tolle, an early innovator in cloud-based software development, explains it all in “How Technology Is Changing the Way We Learn.”

The market we serve demands software delivery to be faster, better, and cheaper. Mike Sowers has some words of wisdom you won’t want to miss in “Achieving Continuous Improvement and Innovation in Software.” In every issue, we include an article on management or soft skills development. What should you do when one employee impacts the effectiveness of an entire team? Software managers are going to want to read Andy Kaufman’s “You Get What You Tolerate.”

We value your feedback. Let us and our authors know what you think of the articles by leaving your comments. I hope you enjoy reading this issue as much as we enjoy working with these wonderful authors.



Ken Whitaker

kwhitaker@techwell.com

Twitter: @Software_Maniac



FOLLOW US



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
Jamie Borders
jborders.com

Advertising

SALES CONSULTANTS
Daryll Paiva
Kim Trott

PRODUCTION COORDINATOR
Alex Dinney

Marketing

MARKETING MANAGER
Cristy Bird

MARKETING ASSISTANT
Allison Scholz

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, Ste. 400
Orange Park, FL 32073



CONTRIBUTORS



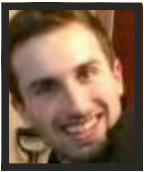
Dewey Hou has worked as a software engineer, project manager, and technical product manager prior to his current role leading product development at TechSmith. Dewey works with teams to improve software development practices in quality assurance, user experience design, technical documentation, project management, and build/release engineering. You can contact Dewey at d.hou@techsmith.com.



Andy Kaufman, the voice of the Institute for Leadership Excellence & Development, Inc., helps organizations around the world improve their ability to deliver projects and lead teams. He has more than twenty-five years of software development experience and is the author of three books. As the host of *The People and Projects Podcast*, Andy provides insights that help listeners lead people to deliver projects. Andy can be reached at andy@i-leadonline.com.



Paul E. McMahon is principal consultant at PEM Systems, where he coaches teams on practical agility and process maturity approaches. Paul released his fifth book, *It's All Upside Down: What I've Learned about Software Development and Why It Seems Opposite to Everything I Was Taught*. It should come as no surprise that his book, like his *Better Software* article, includes true upside-down success stories. When he isn't coaching, you can find Paul running (including the completion of twelve Boston Marathons). Reach Paul at pemcmahon@acm.org.



Josiah Renaudin is a longtime freelancer in the tech industry and is now a web-content producer and writer for TechWell, StickyMinds.com, and *Better Software* magazine. He wrote for popular video game journalism websites like GameSpot, IGN, and Paste Magazine, and now acts as an editor for an indie project published by Sony Santa Monica. 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.



With more than thirty-two years of experience, **Jim Schiel** has worked in software development in highly regulated industries. Jim transitioned Siemens Medical, an organization of more than a thousand developers, from waterfall to agile practices. The author of several books, including *Enterprise-Scale Agile Software Development*, he helps transform organizations worldwide to improve development productivity and product quality. Reach Jim at jim@artisansoftwareconsulting.com.



Mike Sowers has more than twenty-five years of practical experience as a quality and test leader of internationally distributed test teams across multiple industries. He is a senior consultant who works with large and small organizations to improve their software development, testing, and delivery approaches. He has worked with companies including Fidelity Investments, PepsiCo, FedEx, Southwest Airlines, Wells Fargo, and Lockheed to improve software quality, reduce time to market, and decrease costs. Reach Mike at msowers@techwell.com.



Troy Tolle is a leading visionary and champion of cloud computing, used in redefining the future of learning. After leaving academia, Troy cofounded Infinity Learning Solutions and built its DigitalChalk flagship product. As a thought leader in enterprise technology, he is frequently called upon by cloud providers for technical guidance. Troy speaks at both public and private gatherings on various topics including education, cloud computing, and leadership. Contact Troy at ttolle@digitalchalk.com.



Achieving Continuous Improvement and Innovation in Software

THERE ARE SEVEN KEY TIPS AND TECHNIQUES THAT CAN MAKE A HUGE DIFFERENCE IN YOUR SOFTWARE DEVELOPMENT TEAM'S EFFICIENCY.

by **Mike Sowers** | msowers@techwell.com

There is no end to the challenges we face in delivering better quality software quickly to market at reasonable costs. We must secure a firm understanding of user needs, select the appropriate technical design, and develop and test its functionality—all while addressing stated and implied nonfunctional characteristics, striving to develop the right code, employing the latest stable technology, and supporting products that work on multiple platforms. The process depends on retaining the best team with the technical skill sets necessary to get it all done.

Getting Software Done Right

My belief is that getting software done right is a societal requirement. [1] The deployment of software has reached utility status—we rely on it just as we rely on water and electricity. Yet many of the organizations I work with at conference events, in training sessions, and during mentoring and coaching engagements continue to struggle to deliver software with the right functionality, the right level of quality, at the right time, and at the right cost.

We've made some great progress evolving our design, development, testing, and delivery methods. These include:

- Improving technical skill sets
- Employing new or improved programming languages
- Creating new integrated development environments
- Implementing testing tools (both commercial and open source)
- Using DevOps approaches, including virtualization, continuous integration, and continuous deployment

But even with these successes, I think there are ways the software community can more consistently develop and deliver functionally sufficient software to our users quickly and cost-effectively.

Delivering Software Faster, Better, Cheaper

Here are seven approaches that should be the catalyst for better, faster, and cheaper software development. None are to be taken in isolation but knitted together, as appropriate.

Automate with intention: How do we produce better cars more quickly? We automate many parts of the design and manufacturing processes. During a TechWell STAR software testing conference keynote, industry technology analyst Theresa Lanowitz suggested that the next opportunity in delivering better software is automating across the entire lifecycle. I interpret this to mean not just a random, uncoordinated approach to automation, but an orchestrated, integrated, and end-to-end automation approach. The opportunities for automation span the entire value chain.

All components associated with DevOps allow us to better manage risk and accelerate features to market.

Employ modeling and prototyping: A picture is worth many user stories. Prototypes are an excellent way to encourage feedback to determine whether features meet user needs. Collaborative modeling tools, such as sticky notes, and computer-aided tools, such as mind maps, UML, design patterns, business process modeling, and agile model-driven development

(AMDD), help the entire team and the stakeholders become aligned.

Embrace continuous quality engineering: Delivering better software with less rework demands built-in quality engineering practices at every step. Continuous improvement advocates using the plan-do-check-act cycle. Creating the tests before the code is created in test-driven development is a beneficial approach to ensuring continuous quality. The same is true for acceptance test-driven development, behavior-driven development, exploratory testing, and continuous testing.

One day, maybe we'll be able to just think about some new software capability and a machine will develop, test, and deploy it for us.

Beginning with product design, the team should always incorporate quality engineering practices. The whole team benefits by emphasizing quality from concept to delivery.

Leverage business intelligence: The more information we have, the better decisions we make. There is plenty of information available about development practices and the products created using these approaches as business intelligence capabilities mature. Pulling integrated information from multiple data sources encourages a data-driven culture, resulting in better decisions. Further, we have more information available on usage patterns, user experiences, application stability, and application performance operating in a variety of environments.

Think packaging, reuse, and integration: As product lines mature, strategies such as consolidation, reuse, repurposing, and higher levels of integration allow producers to remain competitive. Wrapping up software in a complete file system with everything it needs to run ensures that it will always run the same, regardless of the environment.

Containers allow us to do just that, helping us isolate our code from the operating environment. Repurposing proven code increases productivity, improves quality, and reduces development time. The aggregation of features, applications, or subsystems may expand functionality, improve reliability, and drive down maintenance costs, similar in scale to what the chip industry has achieved in increasing hardware component densities.

Implement decomposition and virtualization: Decomposing an application into small components encourages a more modular design that should allow multiple teams to work more independently and smaller services to be more easily refactored. Emulating the behavior of components still under development lets others continue with development and testing.

Drive innovation: Most computer hardware provides the ability to self-test and report diagnostics. Using tools like JUnit, tests can be created for software components, much like the basic input/output system does when a PC powers up. Martin Fowler calls this “simultaneously building a bug detector” while developing software. [2] Extending this concept a bit further, how about software smartbots constantly checking the health of computer services, applications, and systems?

Another opportunity to innovate is with automatic programming. Rather than a developer programming a computer with a specific set of instructions, machine learning offers a paradigm shift, with the computer iteratively learning from data without being explicitly programmed to do so.

The Real Winner Is Software Development Efficiency

Using any one of these approaches is beneficial, but taking advantage of more than one provides more opportunity for improvement.

Let's assume user experience is being monitored and analyzed in production on each software feature of an enterprise software app. We could leverage feature usage data to determine the priority of those features that should be enhanced first, or which automated acceptance tests to run during our work in process release. One day, maybe we'll be able to just *think* about some new software capability and a machine will develop, test, and deploy it for us.

Until then, these approaches can help accelerate your journey in delivering faster, better, and cheaper software. **[BSM]**

CLICK FOR THIS STORY'S **REFERENCES**

WANTED! A FEW GREAT WRITERS



TechWell is always looking for authors interested in getting their thoughts published in *Better Software*, a leading online magazine focused on 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
- DevOps
- Project and people management
- Configuration management

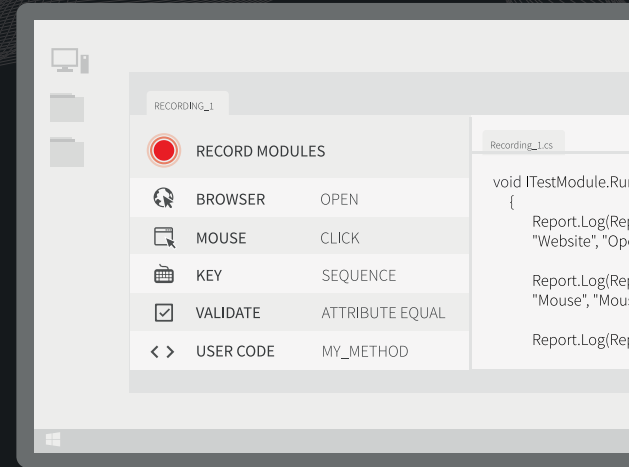
I'm looking forward to hearing from you!

Ken Whitaker

Editor, *Better Software* magazine | kwhitaker@techwell.com

All-in-One Test Automation

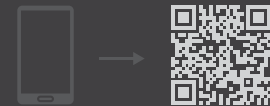
-  Any Technology
-  Seamless Integration
-  Broad Acceptance
-  Robust Automation
-  Quick ROI



Now with Selenium Integration



- Access Selenium with the Ranorex automation framework
- Address Selenium pain points
- Web testing across all major platforms and browsers
- For testers and developers



www.ranorex.com/try-now

INTERVIEW WITH AN EXPERT

“The companies that have embraced DevOps are making a difference. They’re able to spend more time on product development and innovation, and they don’t have to have large groups that are only operations focused.”

“There still is manual testing, but it’s going to be more exploratory testing, ad hoc testing. Having dedicated manual testers is going away.”

“When you start with testing, it really creates energy around this movement, around what you’re doing in quality, and then that propels even further to the other DevOps capabilities.”

“**Focus on flow. The whole purpose of DevOps is to be able to deliver high-quality software production early and often.**”

Adam Auerbach

Years in Industry: **17**

Email: **Adam.Auerbach@lfg.com**

Interviewed by: **Josiah Renaudin**

Email: **jrenaudin@techwell.com**

“It is unfair in some regards that people are picking on testing again, but by embracing it, you really have an opportunity to be the driving force at your company for this [DevOps] movement.”

“When you’re doing manual testing, you’re becoming a bottleneck, and then you’re getting squeezed and pushed.”

“We have to have tools to be able to procure our data in an automated fashion that can be hooked into our pipeline.”

“Testers have the opportunity to, instead of being at the tail end of a process, to be there from the very beginning.”

[CLICK HERE FOR THE FULL INTERVIEW >>](#)

The wait for test data is over.

GET OUT OF QUEUE

DEPTIX

WE LOOK FORWARD TO SEEING YOU IN ORLANDO THIS FALL!

Special Offer for Better Software Subscribers:

Register by 9/8/17 with promo code BSME to save up to an additional \$600 off*

Benefit from a custom week of learning and discovery through all aspects of the development lifecycle with:

- Comprehensive tutorials
- Exceptional concurrent sessions
- Inspiring keynotes
- Pre-conference training and certification classes
- Networking activities
- The Expo
- And more

2017

Your Name

Delegate

Agile Dev
Better Software
DevOps **EAST**

A TECHWELL EVENT

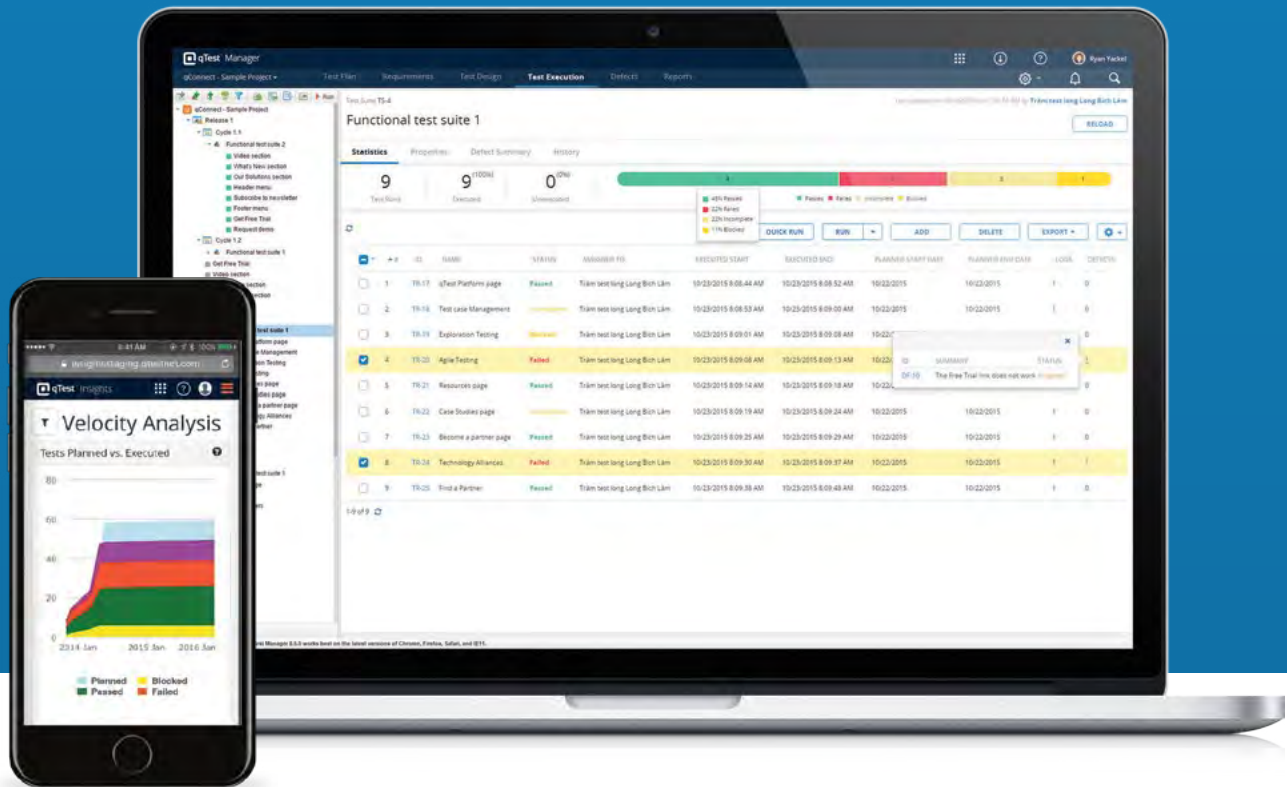


Nov. 5-10, 2017
Orlando, FL
Hilton Orlando
Lake Buena Vista

BSCEAST.TECHWELL.COM

*Discount valid on packages over \$600

WATERFALL, AGILE, OR DEVOPS? YES.



QASymphony is the creator of qTest —
A more efficient software testing platform
that any team can use with any methodology.

Until now you've been forced to choose between either the same old legacy tool from a QA dinosaur or a promising new tool from some startup that can't support enterprise-scale needs.

In the real world, you don't have the luxury of choosing between innovation and scale. That's why leading enterprises like Cisco, Salesforce, Barclays and Amazon rely on qTest by QASymphony. qTest helps hundreds of companies across the globe test smarter, test seamlessly and test at scale. Finally, a testing solution built for the real world.

 **QASymphony**
HELLO, REAL WORLD™

Start a **Free Trial** Today at
www.qasymphony.com

10

Lessons Learned in Cross-Platform Development

BY DEWEY HOU

I help lead a team that is responsible for the development and release of software apps for macOS and Windows. You might assume it would be simple to share universally valuable lessons I learned on our journey of cross-platform development. This isn't an easy task and you can't always guarantee a positive outcome. It is difficult to design and build software apps for different platforms.

There is always a risk that the cost of development won't meet return on investment goals or that the end-user dislikes how the app works on their platform. Figure 1 shows screen captures of our video-editing software, Camtasia, on Windows and on macOS.

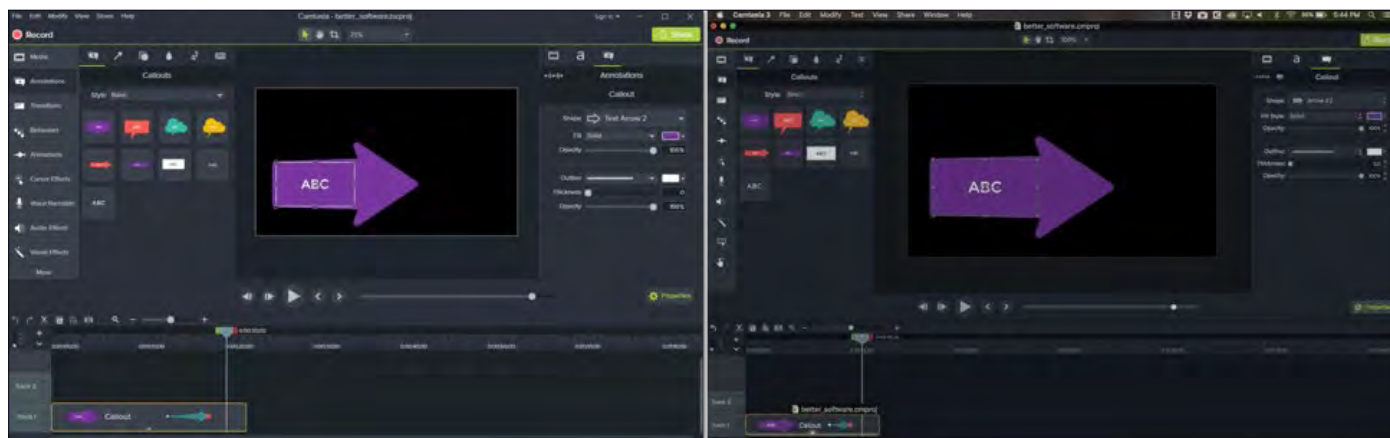


Figure 1: Windows and macOS user interface comparison

These screen captures are a sample of many more you can view on our website. [1]

As you can see, the same program can look different on a computer running macOS versus a PC running Windows, and this presents challenges for a development team. Most software companies dedicate separate teams to work on each target operating system platforms. This can be quite costly and isn't the most efficient approach. After many twists and turns during the latest release of our Camtasia app, several lessons emerged in the areas of business, users, technology, people, and process.

I'd like to share some of what I've learned in developing software applications for Windows and on macOS.

Multiplatform Benefits
Higher percentage of macOS users in future target market segments
Project interoperability is becoming the norm with work groups in corporate environments
License model can be simplified, allowing users to run either version (Windows or macOS)
Multiplatform support is a competitive advantage in the marketplace
Reduce training costs for organizations

Table 1: Identification of cross-platform business benefits

How Business Drives Development

LESSON 1: CONSIDER ECONOMIC FACTORS BEFORE INVESTING

Originally, our app only ran on Windows. Supporting macOS required us to justify the investment to executives. It's a good idea to detail both the business and customer-facing benefits for building an app on a new platform (see table 1).

You must validate the business economics before moving forward. Expanding to a new platform such as macOS merely for technology's sake is not a justification.

Cross-platform development is a nonstarter if the project doesn't make business sense.

LESSON 2: ADD GUIDING PRINCIPLES TO YOUR BUSINESS PLAN

Once you have the green light to develop apps for a new platform, how similar should the apps be? In my experience, it is not practical to require all features and functionality to be identical on both platforms. There are always exceptions, so you better identify some guidelines. Without guiding principles, development costs and timelines can quickly get out of control.

By defining requirements from a business perspective, you can usually predict cross-platform business outcomes. Start by identifying the areas of the app that require platform parity. Separate these areas into priority levels, as shown in table 2.

It is hard work to vet feature requirements—and even more so on two platforms.

The User Experience Influence

LESSON 3: LEVERAGE MULTIPLE PLATFORMS TO TEST PROTOTYPES

Understanding user needs is critical to software development. It is essential to come up with a user interface (UI) design that meets users' needs on both platforms.

Priority Level	Feature	Functionality
Priority 1: Strict parity	Data model/project compatibility	You must always be able to open a project created by the app running on the other platform.
Priority 2: Strict parity with some exceptions	Rendering/pixel-level parity	The content of a project looks/renders the same on the other platform.
Priority 3: Parity when possible	User interface parity	The apps share the same look and layout. The Windows user should feel comfortable using the macOS app (and vice versa).
Priority 4: No parity	Platform-dependent look and feel	Adhere to platform-specific conventions and norms or ignore if it doesn't make business or technical sense.

Table 2: Guiding principles based on cross-platform business priorities

Our first version of Camtasia for macOS came years after the first Windows version. As a result, it would not achieve parity from the start. We took advantage of this fact and created an entirely new design for macOS. Our goal was to make a better product, given years of lessons learned from the Windows version. This approach allows your development team and customers to discover what works best.

Take advantage of the fact that you have two platform sandboxes to play in, and test UI designs. The design could leapfrog ahead on macOS and gain validation before incorporating changes on the Windows side. But introduce changes strategically, as existing customers find it difficult to accept drastic changes to the UI.

Internally, platform biases will be another significant factor during UI design debates. This phased delivery approach takes time, and upper management and your customers must recognize that cross-platform parity isn't going to happen overnight.

Test out designs on one platform to help resolve these disputes and arrive at the best cross-platform design.

LESSON 4: PROCEED WITH CAUTION USING CROSS-PLATFORM UI TOOLKITS

Using a cross-platform UI toolkit seems like a no-brainer. However, the promise of write-once-run-anywhere comes at a cost.

We learned about some issues only after moving ahead with one such toolkit. A few of the downsides included suboptimal user experience, decreased development velocity, poor performance, and inability to leverage the latest platform innovations. Using a toolkit might have been great for our developers, but these negatives would not have been accepted by our customers. If our macOS app doesn't appeal to the Mac users, they'll use something else.

Creating a superior user experience was central to our customer value proposition. The UI toolkit didn't deliver the look or feel that we wanted. It also didn't provide the level of performance needed by a video-editing app. There was an issue not only with learning the toolkit but also with finding developers experienced in programming with that toolkit.

As a result, development velocity slowed dramatically. The learning curve was just too great. To compound the issue, developers still had to work with the details of the native Windows or macOS platforms.

You are dependent on the release schedule of a third-party toolkit for critical bug fixes and support for new features. We found it frustrating to attempt to leverage new platform-specific features. For example, Windows Presentation Foundation (WPF) would not have been available with a third-party toolkit.

What should have been a positive ended up putting us at a competitive and market disadvantage. In the end, we decided to abandon the UI toolkit.

Of course, these issues may not apply to your situation. But for us, native platform development for the UI became necessary.

LESSON 5: BE PRODUCT-CENTRIC INSTEAD OF PLATFORM-SPECIFIC

Balancing platform-specific conventions and platform independence has its challenges. It doesn't pay to develop a platform-independent app that is viewed as not usable or inconsistent to what a Windows or macOS user expects.

When weighing your options in cross-platform development, enumerate the differences that will be problematic for the app's ability to be platform-independent (see table 3).



Once you have identified platform differences, establish a consistent way your app will handle them. In some cases, staying true to platform-specific conventions makes sense.

Users who are familiar with the platform will expect to find the functionality available. In other situations, take an approach that will retain similar functionality but can be implemented in a way that works well for your app. For example, we created a cross-platform solution for presenting caption text that renders identically on both platforms.

We kept to our guidelines by creating an app and interface that was unique to itself rather than rigidly following a specific platform.

approach, including only functionality that exists on both platforms. For us, it must be standard C++ 14 or earlier to avoid C++ compiler differences. Even then, occasionally problems do come up. As a rule, the newer features of C++ tend to be less compatible than, say, C++ 98 code. Frustrating as it may sound, you don't learn which features may result in compiler compatibility issues until you try to share code. Compatibility issues can be troublesome to deal with, as the following for each loops shows.

```
MSVC allows the syntax:
    for each ( auto item in container )
Whereas, standard C++ requires this syntax:
    for ( auto item : container )
```

Area	Windows	macOS
File Support	MOV files not supported WMV files supported Vector graphics resizing in PDFs not supported MP4 file decoding differences (ex: different lengths reported, which causes project compatibility issues)	MOV files supported WMV files not supported Vector graphics in PDF can resize to any dimension MP4 file decoding differences
User Interface shortcuts	Favors right-click functionality Copy, Paste: Ctrl+C, Ctrl+V Product-specific shortcuts are the same	Favors keyboard shortcuts Copy, Paste: Cmd+C, Cmd+V Product-specific shortcuts are the same
Tooltip support	Large, verbose tooltips	Minimal text allowed
Text and naming conventions	Sentence case	Title case
Typeface/Fonts	Support of Windows typefaces and fonts. Text also renders differently on both platforms	Support of macOS typefaces and fonts. Text also renders differently on both platforms
Media effects	Better support for video transitions	Better support for video and audio effects

Table 3: Identification of platform differences impacting app portability

Selecting the Best Technology

Selecting the right tools for the job is another factor in a successful cross-platform product development effort. Optimal choices enable great possibilities, while suboptimal ones create technical debt that hinders your development.

LESSON 6: SELECT A CROSS-PLATFORM LANGUAGE THAT FITS YOUR SITUATION

Choosing a cross-platform programming language is key. It is an essential practice to maximize the amount of code you can share between different platforms. Language selection depends on business needs, the type of app, capabilities the language must provide, existing codebase, and your development team's skills.

Our app manipulates large multimedia files and requires low-level system access to hardware and the graphics processing unit (GPU) in order to achieve a high degree of performance.

Due to our UI requirements, we knew our app would use cross-platform code and platform-specific code. The team also was already very familiar with C++, so given these factors, it became our cross-platform language of choice.

Cross-platform code must take a least common denominator

Consequently, compiler syntax issues often result in writing more complex shared code. During development, we ran into compiler interoperability issues that we had to work around.

In order for the unmanaged C++ code to talk with the managed C# runtime under Windows, there was a need for an interop layer. Writing the interop layer slowed down developers and introduced performance issues that should have been identified sooner in our development cycle.

“Language selection depends on business needs, the type of app, capabilities the language must provide, existing codebase, and your development team’s skills.”

Fortunately, we were able to focus developers specifically on this issue and were able to build out our interop layer and tools. There are no such interoperability issues between C++ and Objective-C on macOS.



LESSON 7: USE A SINGLE SOURCE CONTROL SYSTEM TO SHARE CODE BETWEEN PLATFORMS

This lesson seems obvious, but we started from the Windows platform and had significant investments in Windows-specific tools. Keeping software changes backed up as versions has always been a high priority.

Ten years ago when we started macOS development, there wasn't a great cross-platform choice that met the needs of both Windows and macOS developers. That resulted in using a separate source control system for each platform for several years. It became a nightmare to synchronize the sharing of code between platforms.

The solution came over time by moving all our code to Git. It was a significant effort and cost to migrate our code, as was training the staff to manage source control. If you can, keep cross-platform code in a single separate repository. For code that changes often, reference this repository as Git submodules under the macOS and Windows codebases. Use NuGet packages for code that changes infrequently. Development teams on both platforms can check out, modify, and merge code with ease. Utilizing a single source code control system significantly improves overall team velocity.

LESSON 8: MAINTAIN A SOUND ARCHITECTURE

You become aware of the complexity of cross-platform development once you begin making decisions about how to write shared code. The development team needs to be intentional in creating and maintaining a solid architecture. Early on, we invested in research spikes to learn and experiment with the best ways to structure the code.

In creating a sound architecture, I strongly suggest starting with the business goals discussed in lesson 2. Business outcomes should always drive an app's architecture. Use proven design patterns, such as model-view-controller (MVC), and a good data model to support cross-platform sharing of project files.

Lay a good foundation from the beginning to keep your app resilient and extensible years into the future.

But creating a solid architecture is not enough. The other requirement is maintaining its integrity. To do this, set up automated unit tests around all shared library code. It should be a hard rule that no new code can be checked in without updating and maintaining these tests. Unit tests should keep the data model bug-free.

A related side lesson is to focus on performance testing up front. When we were developing Camtasia, the Windows version was rebuilt in WPF, and due to implementation issues, our UI couldn't keep

up. Profile code to establish a baseline of UI responsiveness. This helps catch performance issues early before code complexity increases over the course of development.

People and Process

For anything to succeed, you need smart, motivated people who have the right capabilities and experience. Technology choices often depend on the skills your people have.

LESSON 9: PRESERVE KNOWLEDGE CONTINUITY ON YOUR TEAM

Creating a team that can withstand disruptions is no small feat. Hiring and retaining top talent is only the first step. Staff needs to embody the positive culture you want on your team. The willingness to share knowledge and help others for the good of the product and customer.

“Business outcomes should always drive an app's architecture. Lay a good foundation from the beginning to keep your app resilient and extensible years into the future.”

One approach that worked for our department was to put Windows and macOS software engineers on a single product team. This increased cross-platform collaboration and encouraged more teamwork. Allow the larger product team to self-organize into smaller pods focused on getting specific work done. When developing a feature that needs to exist on both platforms, pair one Windows and one macOS developer.

Lead Role	Who Usually Fills the Role	Area of Accountability
Product lead	Technical product manager	Owens decisions on feature backlog priority and requirements
Technical lead	Senior software engineer	Owens decisions on implementation. Each platform should have a technical lead
Delivery lead	Project manager	Owens decisions on team process and schedule
UX lead	Senior UX designer	Owens decisions on the overall user experience across both platforms

Table 4: Leadership roles on a development team

This yields better code and better product decisions. Avoid excessive churn of the pod team members by keeping the teams together longer. While a certain amount of change is healthy, it is a mistake to move people like interchangeable cogs. Continuity of staff will preserve product development knowledge.

LESSON 10: LEADERSHIP AND ACCOUNTABILITY ARE MUST-HAVES

Product development is often contentious due to passionate team members debating what they think is best. On a cross-platform project, discussions can escalate into arguments due to platform bias. Leaders are responsible for creating an environment where teams can do their best work.

Create clear accountability for specific individuals to resolve these issues. Define and establish leadership roles on a team to mitigate the inevitable staff changes over the course of the product release, as shown in table 4. You will cause unnecessary turmoil if you lack any of these leads during critical periods in the product release.

Defining a lead role is not about a new job title, but about making an impact on the success of a project. For example, a technical lead for shared cross-platform code can guide the team to prevent data model corruption. Without that lead, it is likely that poor decisions—or no decisions at all—would be made. The UX lead role is particularly challenging because detailed knowledge of both platforms and user needs are essential.

Adopting Lessons Learned

Your experience in cross-platform development will probably be dramatically different from mine, but these lessons should hold true for any project. I wish you luck as you embark on your journey in developing for multiple platforms. [BSM]

d.hou@techsmith.com

CLICK FOR THIS STORY'S

REFERENCES

NEWSLETTERS FOR EVERY NEED!

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

AGILE CONNECTION™ *To Go*
A TECHWELL COMMUNITY

AgileConnection To Go covers all things agile.

DEVOPS *To Go*
BROUGHT TO YOU BY CMCROSSROADS

DevOps To Go delivers new and relevant DevOps content from CMCrossroads every month.

STICKYMINDS™ *To Go*
A TECHWELL COMMUNITY

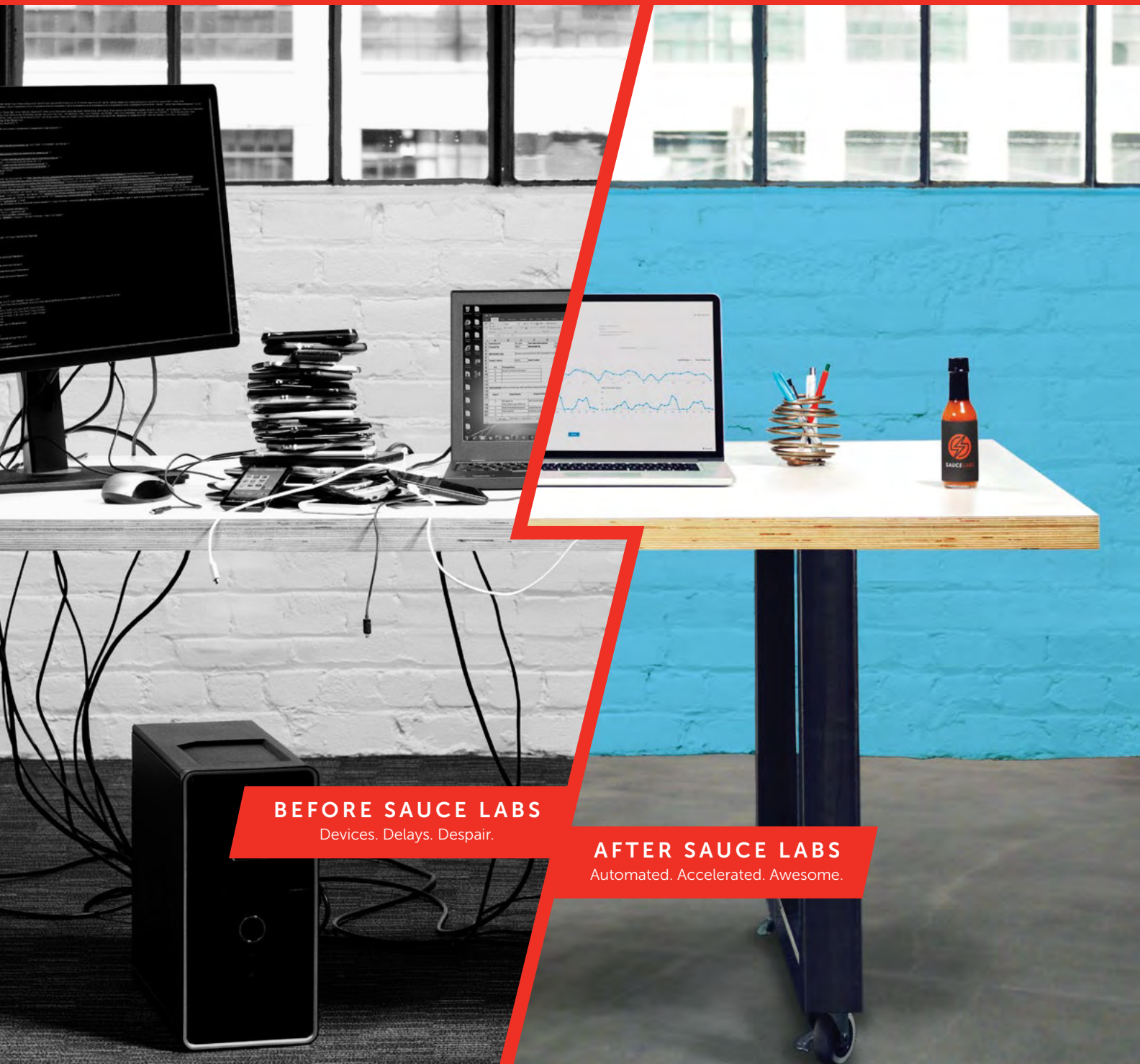
StickyMinds To Go sends you a weekly listing of all the new testing articles added to StickyMinds.

TECHWELL™
INSIGHTS

And, last but not least, **TechWell Insights** features the latest stories from conference speakers, SQE Training partners, and other industry voices.

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

A brief history of web and mobile app testing.



BEFORE SAUCE LABS

Devices. Delays. Despair.

AFTER SAUCE LABS

Automated. Accelerated. Awesome.

Find out how Sauce Labs
can accelerate your testing
to the speed of awesome.

For a demo, please visit saucelabs.com/demo
Email sales@saucelabs.com or call (855) 677-0011 to learn more.



Testing at the speed of awesome.

Is your legacy testing platform holding you back from Agile & DevOps success?



Tricentis Tosca

It's different. It works. We'll prove it.

FREE
TRIAL

tricentis.com/better-software

 TRICENTIS

A man is performing a handstand on a wooden floor against a solid orange background. He is wearing a teal tank top and brown pants. His feet are at the top of the frame, and his hands are on the floor. The title text is overlaid on the right side of the image.

THE POWER OF THINKING UPSIDE DOWN

Paul E.
McMahon

I have found that the best way to coach software development teams may appear to be the opposite of many well-established, long-held software engineering principles. What works in practice often isn't what we have been taught.

This real story demonstrates what I recently discovered works for successful software development teams—even though it may appear to be “upside down” from traditional thinking.

Defining New Processes Up Front Is Old News

Most of us have been taught to define processes and prove them in a pilot environment before using them on a real project. This approach makes sure teams use these processes in a repeatable way before improving them.

This makes sense in theory, but there's a better way. What if you spent very little time defining a process before trying it out on a real project? This is not the way most of us do it, yet I have successfully worked with multiple software development teams to evolve into using new processes during development.

This is not meant to suggest that you shouldn't define your processes and train your people before asking them to use the processes; nor is it meant to suggest you should try out completely unproven ideas on critical projects. Instead, spending very little time defining processes before asking teams to use them might have hidden benefits.

Using this upside-down thinking might not be as risky as you might imagine.

Focus on Strengths and Ignore Weaknesses

Software development organizations usually employ standard processes and procedures. Some can be viewed as strengths and others as weaknesses. Examples of strengths might be the way an organization involves its stakeholders before a product delivery or the way it proactively manages risks. Examples of weaknesses might be a regression test practice that fails to cover critical product functionality, or an inadequate peer review practice. When an organization needs help, outside consultants can be used to help address those specific weaknesses.

There's a better upside down approach. Rather than focusing on a client's weaknesses, it is often a better idea to start a consulting engagement by ignoring known weaknesses. Instead, take the time to understand an organization's strengths. There are two reasons why this might make sense:

- It is very easy to inadvertently damage an organization's strengths when implementing solutions to weaknesses. Learn what those strengths are first.
- Understanding an organization's strengths can often lead to the best solution to fix weaknesses.

If you take the time to look for strengths in an organization before attacking known weaknesses, you could find at least one project that is already working to solve that weakness. This is because any weakness that is an immediate risk to one project is probably equally dangerous to other projects.

Working with developers who are motivated to solve a problem gets you past what often turns out to be the biggest obstacle to a lasting solution. The rest of an organization is more likely to accept a solution proven by one of their own teams rather than an idea from a consultant.

When Just Being Agile Isn't Enough

Let's look at a real example with a client I will call Company X.

Company X is a relatively small organization of less than 100 people providing software to the government. Their parent company was known for requiring heavyweight processes on all of their projects. But when Company X broke off on their own, they wanted to be agile. They intentionally left behind the baggage of their parent company's processes. Becoming agile wasn't the answer. Many of their software releases were late and defect-ridden. Customers were unhappy and threatening to never use their products again. As a result, they brought on board outside consulting help when they realized they had gone too far in dropping defined processes.

A new project needed to be delivered to an already unhappy customer. The consultant seized the opportunity to solve the problem using upside down thinking. Specifically, the consultant gave the team a brief training session on involving key stakeholders and improving communication between developers and testers, and then coached them through the first few months of their project. This was counter to a traditional path of conducting a comprehensive gap analysis, followed by process definition, piloting, and training. That would take months and there was no time.

The result? The team delivered a high-quality, on-time product to their customer.

RATHER THAN FOCUS ON A CLIENT'S WEAKNESSES, IT IS OFTEN A BETTER IDEA TO START A CONSULTING ENGAGEMENT BY IGNORING KNOWN WEAKNESSES.

Rethinking How to Manage Tasks

Company X needed to make changes rapidly to address their pain points of basic task management and testing. Task management was critical because Company X was in constant interrupt mode always reacting to the latest fire-drill in the company. When it came time to release, Company X never had enough time left in the schedule to do adequate testing, which was why their releases were often bug-ridden.

To make matters worse, each release was often focused on changes requested by a specific customer. The development team had a history of failing to conduct adequate regression testing leading to breakage of functionality used by a different customer.

To address these critical pain points as rapidly as possible, the consultant held training sessions on several Scrum-like practices. He emphasized the goal of each practice and how that goal could help Company X with specific pain points.

As a rule of thumb, the team should always conduct sprint reviews with customers. The consultant emphasized that the goal of this practice was to make sure the customer was on board and participated with product acceptance. Because two specific customers used the product differently, the consultant stressed the importance of engaging with both customers.

When it became evident that one of those key customers could not attend the sprint review, the team brainstormed how to meet sprint review goals. As a result, the product owner agreed



to take an early version of the product to get that customer's feedback. This was an innovative solution the team came up with to achieve the real goal of the sprint review practice. Even agile projects can become routine when we forget the why behind certain tasks.

This is upside down thinking in that the team didn't focus on following repeatable steps in a process, such as going through the motions of a sprint review meeting without a key customer. Rather, they focused on what it would take to achieve the real goal, which was key stakeholder involvement and buy-in.

A Unique Approach to Testing

Another identified pain point was testing. Company X didn't have a defined testing process, and too many defects were escaping internal testing only to be found after delivery by the customer. They could have taken the traditional approach of defining a detailed regression test suite and requiring it to be run prior to each release. But given how long it would take to build that suite, the team knew that approach would not solve the immediate quality problems in a reasonable amount of time.

Once again, the team held brainstorming sessions to discuss specific reasons why their testing wasn't working well. First, they agreed they needed to build a full regression test suite, but they couldn't wait until it was done to address this problem. Instead, they immediately started building and using small pieces of the test suite focusing on areas they knew were likely trouble spots.

Along with incrementally building their test suite, they adopted a focused regression testing approach where they selected specific areas to spend extra time testing before each release. This was based on the likelihood of problems, considering past experiences

and known changes in the current release.

They knew that part of their testing problems was caused by poor communication between developers and testers. One tester complained that he couldn't tell what to test from the tickets the developers were completing. After discussing this problem, the developers agreed to start placing better notes in their tickets to make it clear to the testers how to set up the tests and what to look for to ensure the change worked correctly.

Company X could have made placing notes in tickets a required part of the defined testing process, and they could have had their quality group check to make sure that was happening. This is what is often done in many traditional organizations to make sure a process improvement is followed.

But when you take this approach, you are at risk of developers just going through the motions of creating a note in the ticket so they don't receive a non-compliance report from the quality group. Developers can forget the goal is not placing a note in a ticket, but rather improved communication with testers.

The consultant made it clear to the team that placing a note in the ticket is one way—but not the only way—to help achieve the goal. He let the team know that when a developer felt it would be more effective to just go talk to the tester, then they should use that option. Of course, this made it more difficult for the quality group to verify compliance to the practice. It is far easier to verify that a note has been placed in a ticket.

Are you trying to make the job easier for your quality assurance auditors, or do you really want to improve the performance of your software teams?

Putting Upside-Down Thinking to Use

Don't spend a lot of time defining practices first. Rather, coach the team and make them aware of choices to achieve project goals.

In our example, the development organization was able to rapidly put improvements in place leading to reduced latent defects, happier customers, and measurably higher performance. If specific processes had been dictated from the start, the team would not have bought into the more flexible approach. It helped to hear about the positive results achieved on a project in real time by their own teammates.

This doesn't mean that this approach works on every project. There certainly are situations where organizations need to be more prescriptive in the activities, like dealing with life-critical applications.

However, by being less prescriptive with defined processes and coaching teams in how to find innovative solutions during the project lifecycle, software development teams can dramatically increase their likelihood of achieving higher performance results.

This may sound upside down from what many of us have been taught, but in practice it works. **[BSM]**

pemcmahon@acm.org

Apica Performance Script — ~/Desktop


ZebraTester

Apica Performance Script

MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS SQL ASP.NET COFFE LUA SWYFT OBJECTIVE.C RAILS GRADLE JSON DAGGER R
 JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS SQL ASP.NET
 RAILS GRADLE JSON DAGGER REACT ELIXIR GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR N
 IIS SQL ASP.NET COFFE LUA SWYFT OBJECTIVE.C RAILS GRADLE JSON DAGGER REACT ELIXIR GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F
 ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS SQL ASP.NET COFFE LUA SWYFT OBJECTIVE.C RAILS GRADLE JSON DAGGER REACT ELIXIR GO
 GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS SQL ASP.NET COFFE LUA SWYFT OBJECTIVE

The only stack you need, to test your other stacks.


GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS
 RAILS GRADLE JSON DAGGER REACT ELIXIR GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR N
 ASP.NET COFFE LUA SWYFT OBJECTIVE.C RAILS GRADLE JSON DAGGER REACT ELIXIR GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG
 APACHE MYSQL PHP WINDOWS IIS SQL ASP.NET COFFE LUA SWYFT OBJECTIVE.C RAILS GRADLE JSON DAGGER REACT ELIXIR GO ESPRESSO JAVA RUBY
 MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS SQL ASP.NET COFFE LUA SWYFT OBJECTIVE.C RAILS GRADLE JSON DAGGER R



Apica Synthetic

Visualize web, mobile, and API performance.

JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS SQL ASP.NET COFFE LUA SWYFT OBJE
 GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS
 GRADLE JSON DAGGER REACT ELIXIR GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR NODE LI
 COFFE LUA SWYFT OBJECTIVE.C RAILS GRADLE JSON DAGGER REACT ELIXIR GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CL
 MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS SQL ASP.NET COFFE LUA SWYFT OBJECTIVE.C RAILS GRADLE JSON DAGGER R
 JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS SQL ASP.NET COFFE LUA SWYFT OBJE



Apica LoadTest

Load test any website & application at scale.

GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR NODE LINUX APACHE MYSQL PHP WINDOWS IIS
 GRADLE JSON DAGGER REACT ELIXIR GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CLOJUR MONGO EXPRESS ANGULAR NODE LI
 COFFE LUA SWYFT OBJECTIVE.C RAILS GRADLE JSON DAGGER REACT ELIXIR GO ESPRESSO JAVA RUBY SCALA RUST JADE GROOVY F# ERLANG DART CL

~/Desktop/Apica Performance Script* 1:71

Everyone has performance issues.
Apica can find yours before your customers do.

Request a trial at apicasystem.com/start-trial
 Email sales@apicasystem.com or call (310) 776-7540



STAR

CANADA

A TECHWELL EVENT

Oct. 15–20, 2017

Toronto, ON

Hyatt Regency Toronto



WE'VE RESERVED
YOUR SEAT

CONFERENCE TOPICS INCLUDE:

Agile Testing
Mobile Testing
Continuous Integration

Cloud Testing
Performance Testing
DevOps

Test Design
Test Automation
Test Management

RESERVE YOUR SPOT NOW FOR THE BEST PRICING

Better Software subscribers can receive up to
an additional \$200 off with code **BSM**

Learn More: <https://well.tc/sc17bsm>



Fast Track Your App Release

All-in-One Test Automation Suite
For Mobile, Web & Cloud



Save
50% Cost



Reduce
Test Effort by 60%



Achieve 100%
Test Coverage

For a free trail or to schedule a demo
✉ contactus@rapidvaluesolutions.com

🌐 rapidvaluesolutions.com/accurate

[Learn More](#)



10 THINGS YOU MUST DO TO BECOME TRULY AGILE

BY *Jim Schiel*

As an agile coach and consultant, I have one question I ask my clients when I start an engagement: “Is your organization truly agile, or are you just using the right words?” You would think, by this time, that there would be a significant number of truly agile organizations positively changing their environments.

For many software development organizations, being agile is all about doing Scrum, kanban, or Extreme Programming (XP). But even though agile frameworks model agile principles, using these frameworks doesn’t automatically make the organization agile. Agile is not a state of *doing*; it’s a state of *being*.

Being truly agile is clearly stated in the Agile Manifesto as being focused on individuals and interactions, working software, customer collaboration, and responding to change. These values must be built into the organization and consistently put into practice on projects.

So, how do you know if your organization is truly agile or just using the right words? Based on my years of experience, I have created a list of ten things you must do to become truly agile.

1. Focus Business Models on Value

Business models of agile organizations focus less on the efficient delivery of products and services and more on listening to customers, earning their loyalty, anticipating their needs, and determining how to create new customers. Instead of producing more, the agile organization learns to emphasize value and continuously adapting plans.

In 1999, Whirlpool was desperate to improve customer loyalty and implemented a program of innovation from everyone, everywhere. They required a certain amount of innovation in every product development plan.



Whirlpool’s customer loyalty index has risen by 68 percent and their stock has nearly tripled. [1] Whirlpool may not be a software provider, but the example shows how the right business model can positively influence outcomes. Organizations that still create detailed, linear development plans, and then force the team to follow the plan, are missing the boat. The truly agile organization employs agility to focus their business on delivering value with frequent opportunities for feedback and reassessment of their plans.

“INSTEAD OF PRODUCING MORE, THE AGILE ORGANIZATION LEARNS TO EMPHASIZE VALUE AND CONTINUOUSLY ADAPTING PLANS.”

2. Realize That Agility Is a Mindset, Not a Framework

Looking for easy answers and promised gains from agility, management is frequently convinced to invest in sophisticated frameworks to support enterprise-scale agile projects. Instead of teaching how to be agile, these frameworks frequently create structures other than agile that the organization must learn.

In an agile organization, scaling is organic and driven by an internalization of agile principles across the entire company. In many ways, agility becomes an instinctive mindset. Proper scaling occurs when agility is incorporated across the organization and when management processes and practices are reinvented. Development and management teams alike must adopt and use agile principles to get work done.

3. Make Teams Autonomous

A significant degree of productivity in agility comes from teams that have been given the mandate to get a job done and have been given sufficient autonomy to do so. Unfortunately, many teams are surrounded by non-agile management.

This creates friction that can often be seen when teams are told they are autonomous but find management overriding their decisions and plans. They frequently must wait for those decisions to be made based on traditional metrics (e.g., productivity, efficiency, and estimate-to-actual).

The problems created between autonomous teams and non-agile management can be solved by doing two things. First, you must ensure that the management team understands and uses agile.



Second, and much more importantly, you must look at effective external team leadership as a completely different skill set. This requires training management to support the team by building relationships, articulating clear goals, and helping the team make decisions.

4. Work in Small Batches

In all work, complexity and risk go hand in hand and are directly proportional. Usually, the more complexity, the greater the risk. Complex work requires significant effort, and less complex work requires less effort. If complexity results in the team spending more time fixing mistakes, we've gained nothing. Productivity can be improved by breaking down complex work into smaller, less complex tasks.

While doing testing on small batches of work may seem like it adds a lot of overhead, small batches allow organizations to get better by doing something repeatedly and frequently. It has been my experience that tasks should be sliced into small units not to exceed a few days' work. Of course, completed work should still be inspected to ensure it meets the needs of the customer.

5. Aim for Small Teams

Small teams consistently outperform individuals and large teams, especially when a variety of skills and perspectives are needed to complete a project. I recommend no more than eight people on a team. The bigger the team, the harder it is to get everyone together and to reach agreement on decisions. Smaller teams simply work more efficiently.

When creating a small team, the agile organization should keep some key practices in mind.

Teams rely far more on the right mix of skills than the right mix of personalities. Attempts to align compatible personalities using

assessment tools often provides the same result as having management decide on the membership.

Teams form around common goals and frequently need little direction. Once you've focused your team members on a problem and made sure they have what they need to get it done, they'll usually deliver great results.

Teams work better when everything they need is already within their sphere of influence. Scrum, in particular, defines a team as a set of developers that has all the skills necessary to do what is asked of them.

6. Perform Work in a Single-Piece Flow

Championed by Toyota, the concept of single-piece flow processing has revolutionized how the factory floor is run. In single-piece flow, the team completes a single unit, then starts work on the next unit. As a team completes a unit of work (including coding, testing, and documentation), the item should be immediately demonstrable. This gives the team frequent feedback rather than waiting until everything is built before testing anything.

Single-piece flow yields completed, demonstrable software every few days. Mistakes are quickly found and fixed, and costs are lowered dramatically.

To accomplish single-piece flow, teams must abandon traditional approaches where work is done in steps and handed off from one specialist to another. Instead, team members bring their individual skills to bear on engineering a task or solving a key problem. The team works together to simultaneously collaborate on the execution of the work.

7. Work in Short Cycles

While shrinking team size and complexity of work offers true benefits, don't forget to reduce the length of your iterations, too.

I recommend shortening iterations to no more than two weeks. In agile development, the end of an iteration signals an opportunity to evaluate what's been built against customer needs. Shortened iterations improve quality, reduce risk, and reduce project cost. As a result, time to respond to the inevitable defect or incorrect decision is greatly improved.

I frequently ask coders and testers, "After you write some code or some tests, what do you do next?" Invariably, the coders run the application and the testers run their tests. "If the latest code doesn't work, the sooner we find out, the easier it is to fix," they say.

I couldn't agree more. The more often you test your assumptions, the easier it is to fix them. Longer iterations accumulate risk that your customer won't like what you've built. Shorter iterations help you build what your customer really needs.

8. Value Soft Skills

While technical skills are important, soft skills can be more valuable to sustaining high team performance. These skills include the ability to:

- Argue passionately while not attacking
- Share information freely; full transparency is necessary
- Disagree with a decision while fully supporting it
- Make abstract concepts visible and easily understandable

The best team member I've found is someone who has great communication skills, is innovative, and is adaptable to most any situation at hand. While the ability to code in Java or build effective functional specification is important, I'd recommend giving some attention to those soft skills, too.

9. Monitor Debt Closely

While building products or services, organizations tend to incur a significant amount of debt. Debt is a result of the difference between doing something right and doing something poorly. Debt comes in a variety of categories, as shown in table 1.

Debt	How Debt Occurs
Technical debt	Incurred when we build our products with less than the quality the product demands
Decision debt	Incurred by deferring decisions until you're forced into making one
Learning debt	Incurred by deferring learning opportunities until you have to compromise due to lack of skills
Innovation debt	Incurred by doing only what is safe and never trying to include some degree of new functionality in every project

Table 1: Common types of debt taking place in software development

Debt forces the organization to compromise because available choices are limited due to outstanding debt. While organizations are beginning to understand the importance of dealing quickly

with defects in order to limit technical debt, many still don't recognize the need to pay attention to learning opportunities or research opportunities provided during the typical iteration.

To keep your debt under control, keep in mind the following:

Assume that any debt not being actively addressed is probably getting bigger. Take steps to reduce it by planning daily to improve skill sets, decision-making capability, innovation, and motivation.

When problems do occur, take time to identify the root cause and correct the appropriate issue to keep debt from growing. If an organization fails to deliver on time because of product performance issues, the root debt-related cause might be that the organization lacks the skills to do effective performance testing.

10. Be Brave Enough to Experiment and Fail

The world of business is replete with the remains of companies that got comfortable. A business that doesn't move forward is moving backward, and a business or organization that views experimentation as a gamble will probably lose out in the long run.

Sometimes that means you're going to get it wrong. Consider Thomas Edison's experiments with the light bulb. Edison's assistant logged more than 2,700 individual experiments. When asked by a reporter about his many failures, Edison responded, "I now know several thousand things that won't work!" [2]

To innovate, one must experiment and fail. Failure creates amazing possibilities for learning and growth, and must be made part of the organizational culture. A study was performed with Upworthy (not a software development organization) regarding their emphasis on fostering experimentation and divergent thinking. Their culture assumes that a 95 percent failure rate is a sign of a team doing a phenomenal job of experimenting and learning how to get to the right answer. [3] That's an unknown attitude in many organizations, where teams are encouraged to think until they determine the correct answer.

Teams must be permitted to experiment and fail. Where innovation is called for, so is experimentation. Throwing away a thousand lines of good code for the hundred lines that work is a cost-saver in the long run—consider all of those defects the team won't have to fix.

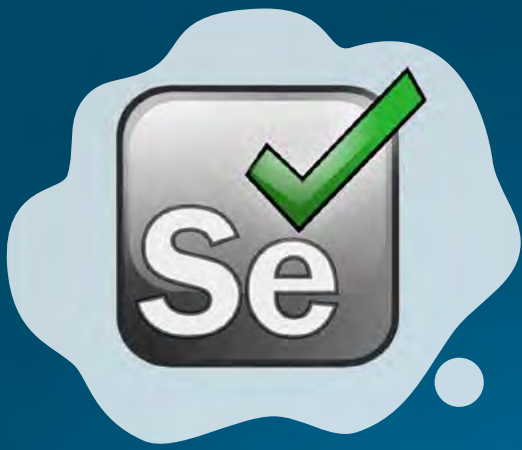
Becoming Truly Agile

Is your organization transforming into an agile one or one that simply uses agile terminology? If you want to be successful in getting the most day-to-day value out of your teams, you've got to be prepared to create new ways of working. To make real agility in your organization, you're going to need to make some real changes.

[BSM]

jim@artisansoftwareconsulting.com

CLICK FOR THIS STORY'S **REFERENCES**



SHHH!

SELENIUM SECRETS & OPEN
SOURCE ADVANCED AUTOMATION

[BIT.LY/GET-SECRETS](http://bit.ly/get-secrets)



 **Perfecto**
perfectomobile.com



Perfecting Software

A COMPLETE IOT SOFTWARE TESTING SOLUTION



“

The importance of software testing and error prevention has risen dramatically, paralleling the continued escalation of software complexity. Parasoft provides developers with the tools and infrastructure necessary to test early and regularly, ensuring quality throughout the software development lifecycle.

Theresa Lanowitz, *voke inc.*

”

STATIC &
DYNAMIC
ANALYSIS

UNIT
TESTING

API, LOAD,
PERFORMANCE, &
SECURITY TESTING

SERVICE
VIRTUALIZATION

Learn more at parasoft.com/iot

LEARN MORE



TRAINING ON YOUR TERMS

Delivered in short lessons with clearly stated learning objectives and regular progress checks, eLearning courses offer a unique approach for software professionals with travel and time constraints. Learn anywhere at your own pace—on your terms.

Get the training you need with SQE Training's eLearning courses: **eSoftware Tester Certification**, **eFoundation for Requirements Development and Management**, **eSelenium 2 WebDriver With Java**, and the new **ePlanning, Architecting, and Implementing Test Automation**. View a no-obligation demo today at sqetraining.com/elearning.

eSoftware Tester Certification—Foundation Level

Accredited training for the ISTQB® Certified Tester—Foundation Level (CTFL) certification. Find out what it takes to be a successful software tester and learn about the relationship of testing to development, test levels, black-box methods, white-box testing, exploratory testing, and more. ISTQB® is the only internationally accepted certification for software testing and has granted more than 500,000 certifications in more than 100 countries around the world.

eFoundation for Requirements Development and Management

Build the foundation you need to successfully develop and manage requirements for business projects and software products in **eFoundation for Requirements Development and Management**. Learn key requirements development and management skills and discover the ways to elicit and document requirements.

eSelenium 2 WebDriver With Java

Selenium WebDriver is the web automation tool of the moment, and Selenium WebDriver skills are in demand. In **eSelenium 2 WebDriver With Java** you will learn real world techniques associated with the Selenium WebDriver API, focusing on the information you need to get productive with Selenium WebDriver. Throughout the course self-learning strategies are emphasized and demonstrated, so that you don't just learn the Selenium WebDriver API in depth, you also learn how to discover more on your own.

ePlanning, Architecting, and Implementing Test Automation

Develop a custom test automation plan and architecture for your organization in **ePlanning, Architecting, and Implementing Test Automation**. Get access to valuable templates you can use to draft your own test automation plan, plus take advantage of one hour of consulting to answer questions and finalize your plan.

SQETRAINING.COM/ELEARNING





©Majdanski - shutterstock.com

**THAT IS
NO ORDINARY
SOFTWARE
TESTER**

CERTIFYING PEOPLE

iSQI 
International Software Quality Institute
#ilovesoftwaretesting



HOW TECHNOLOGY IS CHANGING THE WAY WE LEARN

BY TROY TOLLE

It is not news that the internet has changed how we operate in our daily lives. We stream our shows instead of watching them in real time. We carry the power of the internet in our pockets. We can subscribe to thousands of software-as-a-service (SaaS) offerings to help us with running our businesses, automating our homes, and finding our next meals to cook. Our professional and personal lives are inundated with technology, and it has radically changed how we socialize and conduct business.

However, one area where technology seems to lag is in education. We still rely on traditional models of higher education with full curriculums, transcripts, and grade point averages. The same

is true with corporate training. Most companies still distribute manuals and rely on on-the-job or apprenticeship models for training, with no ability to formally track and measure the effectiveness of these methods.

Thanks to new developments in software, measuring learning in the education space is shifting away from more traditional methods. Training built on classroom lectures with a single grade for a course is being replaced with a more experiential approach to education and the measurement of what a person learned. This new technology is making learning accessible in real time on a wide variety of devices.

The Emergence of the Knowledge Economy

With the rise in services made available through technology, the knowledge economy has emerged. It is essential for businesses to provide easy and fast access to quality educational content to facilitate the growth of their employees. Communities are also finding ways to educate the public without interfering with busy schedules.

The knowledge economy is much more than providing educational materials or affordable internet access. The knowledge economy uses technology growth to share knowledge with a community of coworkers or fellow students, which in turn fosters new ideas and a wealth of new experiences. We are no longer bound by the walls of our offices or classrooms; we can communicate and work with people around the world in real time.

As a result, sharing and generating ideas can take place at a much more rapid pace.

The Evolution of the Digital Learning Experience

Historically, learning has been a face-to-face event between teacher and student. In recent years, however, distance education offered by universities and specialty sites has removed the necessity of in-person learning.

Distance learning and the rise of e-learning began to really take shape in the '90s, when the first learning management system (LMS) was introduced. These software systems were designed to aid in administration, tracking, and reporting for the traditional classroom. At that point, the LMS started replacing the physical classroom with a virtual one. The system became a document repository that housed learning content in file format for download, albeit with minimal interaction capabilities.

In today's marketplace, there are thousands of learning products that meet a wide range of education needs and support many types of content delivery. As platforms have evolved, so has the learning content.

With the increased capabilities offered by an LMS, developers of course authoring tools looked at ways to standardize by packaging materials as a sharable content object reference model (SCORM) file. [1] These zip files contain learning content that adheres to a communication standard for storing and retrieving information about delivery of the content from an LMS. Any content created that conforms to the SCORM specification can be interchanged between one LMS and another.

Though still evolving, SCORM packages are falling out of favor as content providers have begun incorporating more standalone video content into their offerings.

Because high-quality video is readily available on our mobile devices with high-speed access to the cloud, video is a natural medium for sharing and creating content. Done well, video can offer an engaging, effective learning experience by condensing material into bite-sized chunks.

How We Like to Learn Is Rapidly Evolving

There is a notable cultural shift as the next generation of knowledge workers enters the workplace. Learning platforms in businesses will need to adapt to allow more than just chat for collaboration; learners now expect to be able to share user-generated content in the form of documents, pictures, and video. Our mobile-focused society also demands that the systems supporting the learning experience always be available simply by turning to a phone or tablet.

I believe that there are three game-changing shifts taking place: the acceptance of e-learning, the proliferation of intelligent devices, and the emergence of virtual reality.

“LEARNING PLATFORMS IN BUSINESSES WILL NEED TO ADAPT TO ALLOW MORE THAN JUST CHAT FOR COLLABORATION; LEARNERS NOW EXPECT TO BE ABLE TO SHARE USER-GENERATED CONTENT IN THE FORM OF DOCUMENTS, PICTURES, AND VIDEO.”

ACCEPTANCE OF E-LEARNING

The opportunity for e-learning is staggering. According to market research, online corporate training is expected to grow by 13 percent a year over the next decade, with 77 percent of US companies offering technology-based learning. [2] The ability to learn anytime, anywhere makes online education more attractive to students than traditional classrooms. The abundance of quality content has increased employer acceptance of degrees and certifications earned online.

INTERNET OF THINGS AS LEARNING DEVICES

The number of connected devices in the world continues to skyrocket. Gartner predicts there will be more than 20 billion Internet of Things (IoT) devices by 2020. [3] The way we experience the world around us is aided and being chronicled through these everyday devices, and the implications here for learning are huge. If we have a question, we don't have to open a laptop anymore to get an answer—hands-free, voice-controlled devices for our homes, such as the Amazon Echo, mean we can simply ask, “How far are we from the sun?” Our Fitbit can help us understand our bodies and motivate us to get into shape simply by wearing it on our wrist or attached to our clothing.

With the IoT, learning can move from simple communication of knowledge to interactive and engaging ways to learn. IoT allows us to take our learning from the analog to the digital world and to record our experiences, attempts, and accomplishments of tasks. Access to this new data that IoT gives us opens possibilities for new evaluation and measurement. It can help motivate us and bring new revelations to light that can shape our training experiences for the better.

VIRTUAL REALITY AS AN IMMERSIVE EXPERIENCE

Virtual reality and augmented reality are making a splash in the gaming space, and these technologies are fighting to find their place in learning environments, too. Virtual environments are a great way to train someone or immerse them in role-play interactions without putting learners into risky situations or using expensive equipment. Navigating a heart surgery, operating a front loader, flying a fighter jet in a dogfight, or managing a nuclear reactor in a disaster are all possible using virtual and augmented reality devices. The graphics capabilities now are at a level where the experience feels very realistic and can prepare students for the experience in real life.



Tracking the Learning Experience with xAPI

The 70:20:10 model for learning states that 70 percent of our learning takes place through on-the-job experiences, 20 percent from our interactions with peers or mentors, and 10 percent from more traditional educational tasks, such as testing. Until recently, we have been able to measure and track only the 10 percent. If we can't evaluate the entire learning experience, it is impossible to determine what training methods are most effective.

The training and development world tried to collect as much information made possible using LMS technology and in the form

of spreadsheets and files, but this type of data collection quickly becomes unmanageable. The SCORM specification also is limiting because of its focus on communication of data back and forth to the LMS, without a broader specification on the schema of that data. Because of this, LMS and SCORM authoring tool vendors resort to storing information in proprietary formats that severely hamper the ability to report on the details of learning across a variety of courses or implementations.

The Experience API (xAPI) was built to help solve this problem. [4] xAPI greatly improves on SCORM and allows for the recording of all learning experiences—from formal classroom instruction to online interactions. xAPI is a REST API that communicates experiences as a statement, featuring an actor who is participating in the experience, a verb describing the experience, and the object or subject of the experience. These statements are stored in a learning record store (LRS). Figure 1 shows a sample xAPI transaction record describing my watching a video.

Because we experience learning in so many ways, the verbs that can be used in an xAPI statement are vast—participating in meetings, reading blogs, having conversations around the office, watching videos online, taking formal courses delivered through an LMS, conducting simulations, apprenticing, and so much more. The ability to store each of these interactions in a standard format makes the analysis of a person's or group's experiences available for research, comparison, and measurement.

xAPI is still in its infancy, so use of the technology in the learning industry should grow in the coming years as more of our learning experiences become xAPI-enabled. The result of having this increased insight into how people learn will enable companies to train and retain top talent more effectively, provide customized learning experiences for different learning personalities, and let teachers tailor their content based on big data analysis.

Validating the Learning Experience

xAPI and the LRS record a higher percentage of students' total learning in a standard way. One of the great things about a common recording standard is that records can be shared between any LRS implementing the server-side xAPI specification requirements. This means, for example, that as a student moves from college to a job and then from workplace to workplace, all learning records can remain with the individual.

Recorded learning data must be safe and verifiable. There needs to be a way to guarantee accuracy and security.

This is an area of the digital learning experience that hasn't been fully developed yet, but I believe that there is huge potential in using software systems that utilize blockchains. Blockchains are blocks of information that are linked together and stored on a network of distributed, decentralized computers.

This network of distributed devices ensures that security of any block of data is built in. No one person owns the entire system, making it extremely difficult for data to be edited or changed once it has been added to the chain.


```

{
  "id": "2a3a65f3-e19d-437e-910b-18247d13eccc",
  "actor": {
    "name": "Troy Tolle",
    "mbox": "mailto:ttolle@digitalchalk.com",
    "objectType": "Agent"
  },
  "verb": {
    "id": "http://activitystrea.ms/schema/1.0/watch",
    "display": {
      "en": "watched"
    }
  },
  "context": {
    "extensions": {
      "http://id.tincanapi.com/extension/browser-info": {
        "user-agent-header": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:52.0) Gecko/20100101 Firefox/52.0",
        "name": "Firefox",
        "version": "52.0"
      },
      "http://id.tincanapi.com/extension/starting-point": "PT0S",
      "http://id.tincanapi.com/extension/ending-point": "PT3M52.111S"
    },
    "contextActivities": {
      "category": [
        {
          "id": "http://id.tincanapi.com/recipe/video/base/1",
          "definition": {
            "type": "http://id.tincanapi.com/activitytype/recipe"
          },
          "objectType": "Activity"
        }
      ]
    }
  },
  "timestamp": "2017-04-21T15:49:55.317Z",
  "stored": "2017-04-21T15:49:55.317Z",
  "authority": {
    "name": "DigitalChalk",
    "account": {
      "homePage": "https://sandbox.watershedlrs.com",
      "name": "e4f2dedadbf5d4"
    }
  },
  "objectType": "Agent"
},
"version": "1.0.0",
"object": {
  "id": "https://xapi.digitalchalk.com/xapi/video/ff8080814d25224e014d25bdd9890006",
  "definition": {
    "description": {
      "en": "Video for Checkpoints at 15 and 30 - video"
    },
    "type": "http://activitystrea.ms/schema/1.0/video"
  },
  "objectType": "Activity"
}
}

```

Figure 1: Example xAPI statement showing that the author watched a specific segment of a video using Firefox on a Mac

The blocks themselves are linked together using cryptographic algorithms, meaning it would be nearly impossible for the data in the blocks to be modified or forged. Résumés and learning experiences could be instantly validated while still preserving privacy.

Adopting Online Learning

Technology is changing the way people learn, as well as the ways learning experiences are tracked and recorded. Innovation

based on technology advances is opening new doors to experience learning in different ways, track it more precisely, and evaluate its effectiveness. [\[BSM\]](#)

ttolle@digitalchalk.com

CLICK FOR THIS STORY'S **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)



Featuring fresh news and insightful stories about topics important to you, TechWell Insights 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](https://www.techwell.com) for the full stories and more!

Finding the Bottlenecks in the Agile and DevOps Delivery Cycle

By *Tanya Kravtsov*

To achieve incremental software development and continuous feedback, you need to eliminate the tasks that create bottlenecks, which hinder the flow of development. A chain is no stronger than its weakest link, and identifying these “weak links” is a critical step toward achieving agility and increasing efficiency.

[Read More](#)

Managing the Turbulence of Organizational Change

By *Naomi Karten*

In times of major change, particularly organizational change, it's normal for people involved to experience turbulence, including anxiety, anger, or uncertainty. If you're overseeing a change, how you communicate with those affected can significantly decrease—or increase—the duration and intensity of that turbulence.

[Read More](#)

Continuous Testing: New Improvements on an Old Idea

By *Alex Martins*

The concept behind continuous testing is far from new, but what's different now is that software development practices have evolved to a point where developers are embracing testing as part of their responsibilities. Testing is slowly moving from being an “event” to an activity throughout the development lifecycle.

[Read More](#)

Training and Tomorrow's Jobs

By *Pamela Rentz*

Reports vary in predictions about what parts of the workforce will be most affected by automation. How far-fetched is the idea that significant numbers of technology jobs will become irrelevant? How can training and education programs better prepare us for the future?

[Read More](#)

Think Small: Break Down User Stories for Agile Success

By *Mitch Goldstein*

The entire agile team needs to be involved in a continuous process of identifying ways to simplify work, right up until a story is complete. Smaller stories ensure that development work is rapid and trackable. Mitch Goldstein details how to focus on breaking stories down into a more estimable, “digestible” size.

[Read More](#)

Tester Contributions to Scrum Conversations

By *Justin Rohrman*

Scrum is one of the most popular paths to agile, but testers sometimes join this framework as an afterthought and aren't quite sure how they fit into the development flow. Scrum is more than answering three daily questions, and testers are in a position to understand the project better than anyone else on the team.

[Read More](#)

The Future of Testing: Quality Engineers and Specialist Skills

By *Melissa Tondi*

Many testers have opinions about the future of their profession. Melissa Tondi thinks the traditional QA position is moving toward that of a quality engineer—a skilled role using techniques previously thought of as the domain of specialists. If we focus on efficiency, tomorrow's testers can expand their skill sets.

[Read More](#)

The Software World Is Changing—Are You Willing to Change with It?

By *Lee Copeland*

The software landscape is changing. Processes are becoming quicker and leaner, but instead of re-evaluating some of our traditional practices, we sometimes try to make them fit where they don't belong. This holds back continuous improvement. If you want change, you first need to be willing to change.

[Read More](#)

Keeping Your Software Testing Abilities Relevant Today, Tomorrow, and Beyond

By *Sunil Sehgal*

Development and product teams have embraced agile and DevOps. What can testers do to keep up with their development peers? Here are some ideas about what testers can learn, what skills we can add, and what processes we can start doing in order to continue delivering quality today, tomorrow, and further into the future.

[Read More](#)

Engineering Architecture Systems for a Faster Build

By *Abraham Marin-Perez*

In the era of continuous integration and continuous deployment, big applications are creating bloated build pipelines. The problem is when code becomes so entangled that every change impacts large portions of the system, meaning there's a lot to rebuild. If you reshape the code architecture, you can reduce build times.

[Read More](#)

Make Your Security Testing More Agile

By *Alan Crouch*

Security practices traditionally have followed a waterfall model, adding security testing on at the end. Organizations need to coach their security programs and testers to prioritize analysis and risk, much like we do with agile stories, to better incorporate security defects with other feature work along the way.

[Read More](#)

7 Good Project Management Practices for Replacing a Legacy System

By *Payson Hall*

When you need to replace a legacy system quickly, it's tempting to set aside good project management practices and push forward recklessly. But doing so results in delays, cost overrun, and organizational chaos. Take time to understand the problem, plan and estimate the solution, and set up your project for success.

[Read More](#)

The Difference between Managers and Leaders

By *Steve Berczuk*

You often hear managers referred to as leaders, but the two terms are not synonymous. Managers can be leaders, but not always, and there are people who don't have formal management positions who are leaders. Understanding the difference can help people in both roles—and their team members—be more effective.

[Read More](#)

Test Your Data Quality to Increase the Return on Your QA Investment

By *Shauna Ayers*

With the high volume of data coming into your organization, it's important that it be complete, correct, and timely. But considering the velocity at which this data is moving, how do you measure its current quality? You must be able to test it wherever it sits still enough to be viewable, without altering it.

[Read More](#)

10 Strategies to Get the Most out of Attending a Conference

By *Greg Paskal*

Any time you get the opportunity to attend a conference, think of it as a chance to learn and bring some new ideas back to your team and company. It's important to be intentional as you prepare and to know what you want to achieve. Here are ten strategies—and a worksheet—to help you get the most out of the experience.

[Read More](#)

Encouraging Just-In-Time Testing

By *Mukesh Sharma*

When the development landscape is extremely dynamic, a testing effort that is adaptable and flexible with an ability to learn the system and craft scenarios on the go is increasingly important. Testers should be encouraged to be just-in-time testers with the ability to test anything at any time.

[Read More](#)

Building for the Internet of Things Is Great—Just Keep Security in Mind

By *Chris Poulin*

The Internet of Things gives us opportunities to transform everyday life into frictionless interactions between humans and machines. However, that also means the technological attack surface is everything. Makers learning how to build IoT devices must also learn how to build safe, secure, and compliant devices.

[Read More](#)

What You Should Consider to Make the Best Use of Your Collected Data

By *Catherine Cruz Agosto*

We live in a world where data is constantly being recorded. In software, determining the timing of when to use that data is critical to making the most of the information. You should take into account data freshness, the data-gathering processes and any dependencies between them, and when to distribute information.

[Read More](#)



You Get What You Tolerate

IT IS NEVER EASY DEALING WITH DISRUPTIVE AND CONFRONTATIONAL DEVELOPERS—ESPECIALLY WHEN THE PRODUCTIVITY AND WELL-BEING OF THE TEAM ARE AT RISK.

by **Andy Kaufman** | andy@i-leadonline.com

Early in my management career, I had the opportunity to work with a talented developer. I'll call him Sam. Sam was smart, for sure, and he knew it. He also knew he was the only person who deeply understood some critical modules of the software we were developing.

However, Sam struggled with emotional intelligence. He was often oblivious to how his words and negative actions impacted others. If someone disagreed with him, he tended to write them off as stupid, and he could easily lose his temper in meetings. He refused to follow many of our internal processes because they were, in his opinion, worthless. In short, team members and business stakeholders trod lightly around Sam, hoping they wouldn't set him off.

Managers put up with Sam's belligerence over the years because he was technically valuable—we couldn't afford to upset him or risk losing him to a competitor. The company was dealing with a ticking time bomb. After being passed from group to group, Sam landed in my organization. I was the last remaining manager for Sam to work with, and I was a relatively inexperienced manager at that.

What was I to do with Sam?

What We Tolerate

Several years ago I came across a quote that has had a lasting impression on me: "You get what you tolerate." [1] It was used in the context of marriage relationships, but it applies to many areas of what we do as professionals. This is especially true for those of us who lead teams that deliver software projects.

Everyone, especially management, tolerated Sam. But the impact of his behavior was disastrous to team members. Trust quickly eroded when the tsunami of Sam's wrath crashed against someone in his way. Beyond hurt feelings, it hindered productivity and led to increased attrition.

This problem is not unique to software development teams. Sales-oriented organizations often tolerate destructive salesmen as long as they make their numbers, and some organizations tolerate hopelessly demanding and abusive customers because of one reason: "We need the money."

So, what are you tolerating? In my work with software managers around the globe, I often see three areas where we get what we tolerate:

- Team member performance
- Conflict among team members
- Our own careers

"The incorrect perception is that the lower performers on your team may be lazy. It could be they have just gotten into a rut and need to be challenged."

Individual Performance

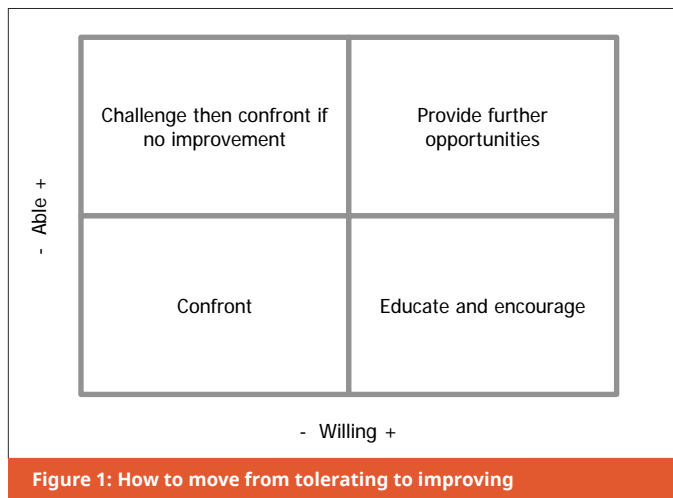
It's easy to start tolerating lackluster performance. Engaged, self-managed teams are a worthy aspiration, but over time, certain team members consistently go above and beyond while others barely carry their own weight.

The incorrect perception is that the lower performers on your team may be lazy. It could be they have just gotten into a rut and need to be challenged. It could be the low performers are sufficiently competent but realized how compensation works at many organizations—they can go above and beyond and get a 3 percent raise while the person who barely performs receives a 2 percent raise.

Regardless, an objective analysis of your team would likely find that you are tolerating lower performance from some team members.

How much does this matter? Researchers of team effectiveness examined the impact of team members who were deadbeats (defined as "withholders of effort"), downers (who tend to "express pessimism, anxiety, insecurity, and irritation"), and jerks (who violate "interpersonal norms of respect"). They concluded that tolerating only one of these negative people can bring down overall team performance by 30 percent to 40 percent. [2]

The willing/able matrix shown in figure 1 can be used to help guide team members to maximum effectiveness.



The goal is for team members to operate in the upper-right quadrant as fully able (competent and skilled) and willing (engaged, motivated, and open to change).

Sam was fully able, in that he was clearly a talented developer. But he had a willing problem. This placed him in the upper-left quadrant of the matrix. Folks in that quadrant need to be challenged to move to the right. This can lead to the manager confronting individuals if their performance doesn't improve.

Conflict and Team Interactions

Beyond individual performance issues, we can also tolerate poorly managed conflict between team members. Teams inevitably encounter conflict, and if you're a manager at all like me, you may not particularly look forward to dealing with it. Yet how your team deals with conflict can be a critical factor in how successfully you deliver your projects. It can be a good thing and lead you to better solutions, or it can be destructive, leading to reduced trust, resentment, and attrition.

Cognitive conflict is the beneficial sort of conflict—the type that wrestles with the ideas and approaches that lead to better outcomes. Cognitive conflict is also necessary, and if you don't have some of it, you could be tolerating something just as insidious: artificial harmony. Affective conflict, on the other hand, is when those interactions go over the line and get personal. Tolerate affective conflict, and it's only a matter of time before you'll see hits to innovation, quality, and overall team performance.

Talk with your team about these types of conflict, and be alert to situations where you begin tolerating affective conflict.

Impact to Career Progression

So, what are you tolerating in your career? Have you grown strangely content working for an organization that treats you more like a resource than a person? Are you tolerating a boss who micromanages you and shoots down ideas without reasonable con-

sideration? Are you settling for a paycheck, having given up on pursuing a path that would be more meaningful but risky?

In his book *Workplace Poker*, [3] author Dan Rust suggests that if you're not happy with the state of your project, the performance of your team, or where you are in your career, there's only one person to blame: yourself. Rust's point is that until you take responsibility for where you are, you won't take responsibility for improving your situation.

The best help I've found for handling career tolerations is a mentor. It's often too difficult to get the perspective we need on our own. Regardless of how formal the relationship is, substantial benefits come from having someone who can help assess *who* we are, *where* we would like to go, and *how* we can get there.

Your Next Move

The willing/able matrix helped me start a long overdue conversation with Sam. I challenged him to find an opportunity to grow his influence at the organization and to improve the team's overall ability to deliver. When Sam pushed back and refused to change, this eventually led to a more difficult conversation. Sam was now confronted with the reality that if he didn't change, it would cost him his job.

I wish I could report that Sam's eyes were opened and he changed his behavior. In the years since, I have seen problem employees respond favorably to challenges and up their game. But that's not how things ended for Sam. I took the step we had all avoided for too long, and Sam was asked to leave the company. This resulted in Sam without a job and our team missing this highly volatile but valued developer.

That seems like a lose-lose ending, but it's not the end of the story.

How long do you think we missed Sam, whom we thought we couldn't live without? Not long. Team morale and productivity improved, and we were still able to complete projects. It turned out to be a positive experience for Sam, too. His wife happened to attend a class I was teaching years later. During a break, she told me that being let go was a terribly difficult time for him, but he could now say it was one of the best things that happened in his career. It was the jolt he needed to make some necessary changes in his life.

You are getting what you tolerate—with your team and your career. Some of these results are benign and unworthy of further thought. But others are holding you and your team back from maximizing true potential. It's your responsibility to remove anything impeding your team.

What are you tolerating? [BSM]

CLICK FOR THIS STORY'S REFERENCES

CAN'T ATTEND A TECHWELL CONFERENCE? WE'VE GOT YOU COVERED!



Check out the TechWell Happenings YouTube Playlists.

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

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

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

LINK TO OUR ADVERTISERS

Agile Dev, Better Software & DevOps East	14
Apica	27
Delphix	13
iSQI	37
Parasoft	35
Perfecto	34
QASymphony	15
QMerty	3
Ranorex	11
RapidValue Solutions	29
Sauce Labs	22
SmartBear	4
SQE Training—eLearning	36
SQE Training—On-Site Training	42
STARCANADA	28
STARWEST	2
Tricentis	23
TurnKey	48

DISPLAY ADVERTISING

advertisingsales@techwell.com

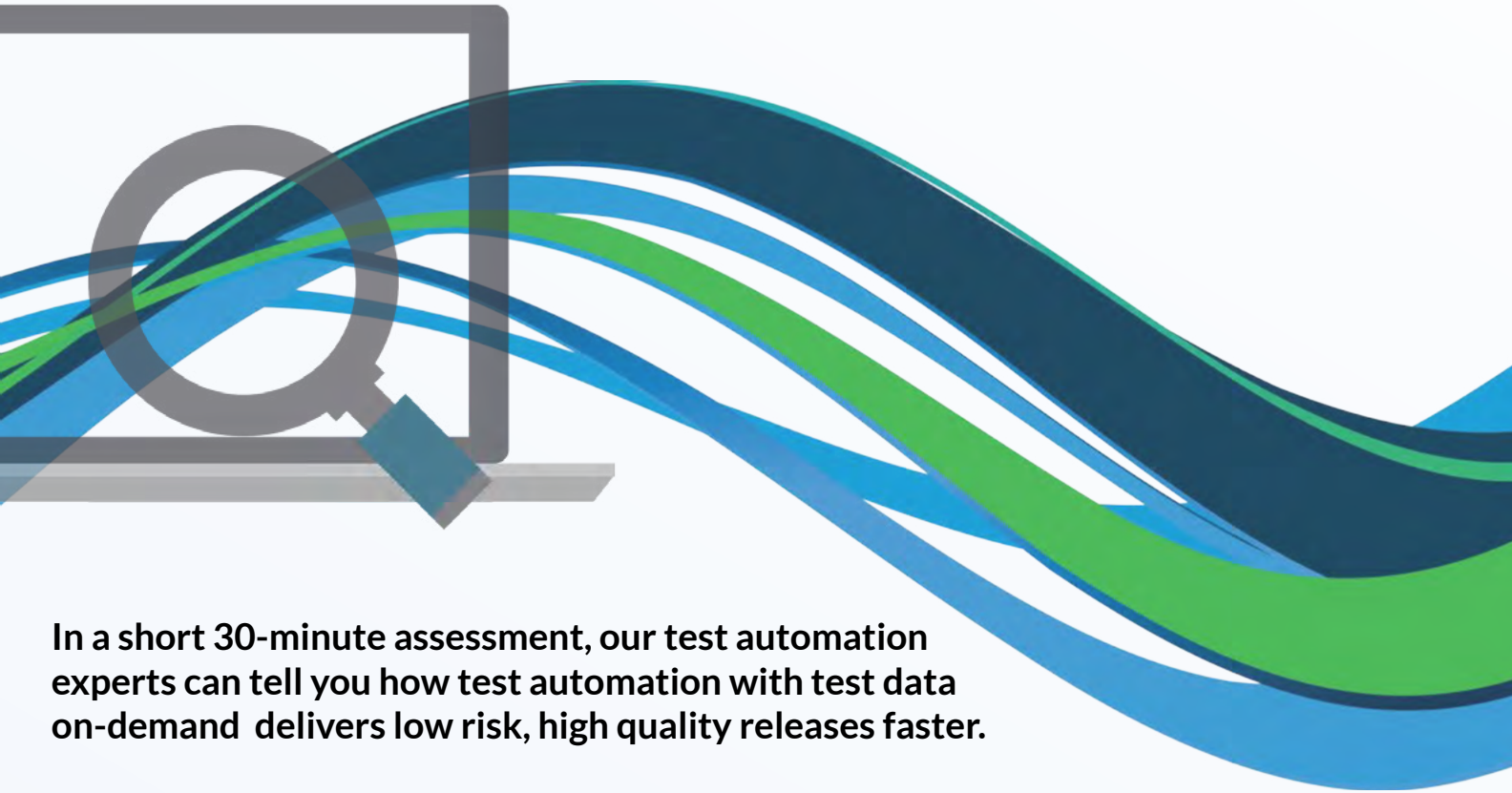
ALL OTHER INQUIRIES

info@bettersoftware.com

Better Software (ISSN: 1553-1929) is published four times per year: January, April, July, and October. Entire contents © 2017 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.



Learn how **ACCELERATE** test cycles,
MITIGATE risk, and **DRIVE DOWN** costs.



In a short 30-minute assessment, our test automation experts can tell you how test automation with test data on-demand delivers low risk, high quality releases faster.

Plus, use our simple ROI Calculator to give you an indication of how much you can save with TurnKey test automation.

Get a 30 Minute FREE, HONEST Assessment
turnkeysolutions.com/free-assessment/ | 844.428.4678

Test Earlier and Faster | Streamline Test Creation | Slash Maintenance Costs Up to 80%