

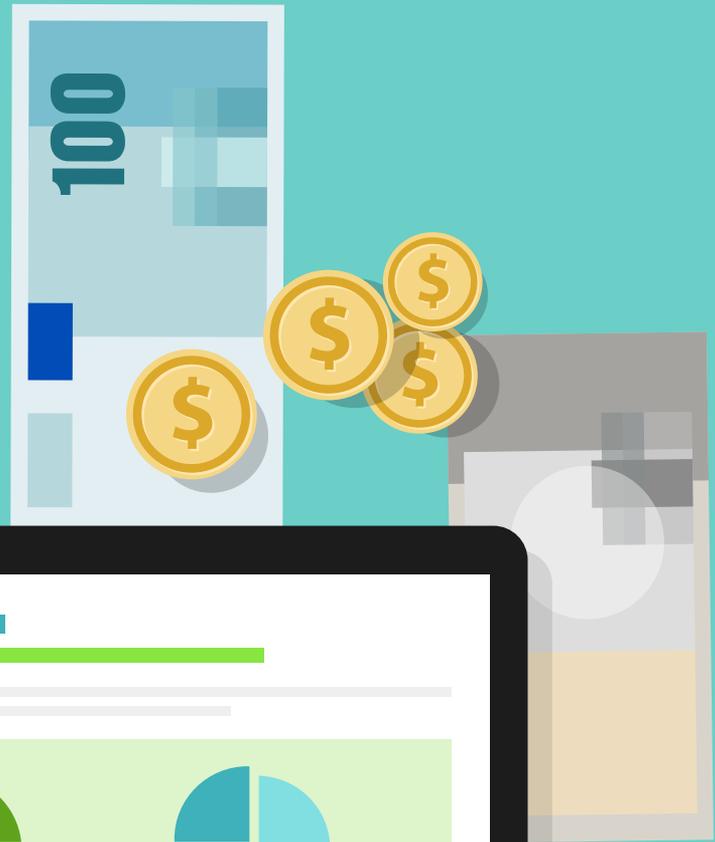
BETTER™ SOFTWARE

A TECHWELL™ PUBLICATION

THE AGILE MINDSET
Avoid simply going through the motions

THE AMAZING CUCUMBER
Automate customer acceptance tests

The Evolution of Software Monetization



All-in-One Test Automation

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

1
License
All Technologies.
All Updates.



www.ranorex.com/try-now

WE LOOK FORWARD TO SEEING YOU IN ORLANDO THIS FALL!

Special Offer for Better Software Subscribers:

Register using promo code BSMCE to save up to an additional \$200 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

2016
Your Name

Delegate

Agile Dev
Better Software
DevOps **EAST**

A TECHWELL EVENT



Nov. 13-18, 2016
Orlando, FL
Hilton Orlando
Lake Buena Vista

BSCEAST.TECHWELL.COM

*Discount valid on packages over \$400



MARK YOUR CALENDARS



Dive Deeper
into Developing
and Testing
Software for
Mobile and The
Internet of Things



Mobile Dev + Test

A TECHWELL EVENT

IoT Dev + Test

A TECHWELL EVENT

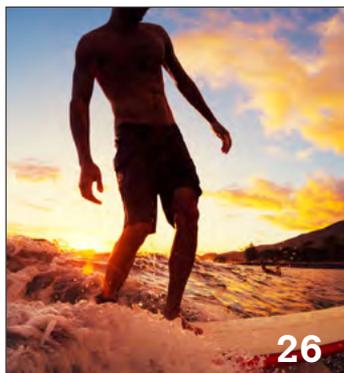
April 24-28, 2017

Westin San Diego | San Diego, CA

<https://well.tc/MobileIoTComingSOON>



CONTENTS



in every issue

Mark Your Calendar	6
Editor's Note	7
Contributors	8
Interview with an Expert	12
TechWell Insights	36
Ad Index	40

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

features

- 14 COVER STORY**
THE EVOLUTION OF SOFTWARE MONETIZATION
End-users are demanding anytime, anywhere access to software apps on their devices. These changes are shifting the way software vendors conduct business. Michael Zunke uses the results of industry surveys to show how software products and services should be licensed.
by Michael Zunke
- 20 THE MINDSET OF THE AGILE DEVELOPER**
Most software development teams these days adopt an agile approach to guide projects through their lifecycle. But, according to Gil Broza, embracing popular practices is not enough. To work effectively in an agile environment, developers must change their mindset.
by Gil Broza
- 26 CONTINUOUS PROCESS IMPROVEMENT USING BALANCE AND FLOW**
Finding a balance between too much and too little process can be quite a challenge. Tom Wessel shows how to apply lean change management and kaizen principles to achieve continuous process improvement. Also, Tom suggests the use of simple metrics to verify that improvements are actually taking place.
by Tom Wessel
- 32 WHAT IS CUCUMBER AND WHY SHOULD I CARE?**
If there ever were a game changer to energize a development team, Cucumber just may be it. An open source tool, Cucumber helps in the running of automated customer acceptance tests. Matt Wynne, a cofounder of Cucumber Limited, delivers a brilliant introduction to this tool.
by Matt Wynne

columns

- 9 TECHNICALLY SPEAKING**
USE DEVOPS TO DRIVE YOUR AGILE ALM
Successful DevOps operations assume that your team has the ability to adjust with constant change. To succeed at continuous integration and deployment, a comprehensive agile plan is needed. Bob Aiello and Leslie Sachs identify four critical success factors that you should employ.
by Bob Aiello and Leslie Sachs
- 38 THE LAST WORD**
WHAT IF SOMEONE STEALS YOUR CODE?
Bob Zeidman, an expert in software forensics, provides a great overview of how to protect your software from predators. You'll learn the difference between copyrights, trade secrets, and patents.
by Bob Zeidman

MARK YOUR CALENDAR

training weeks

Testing Training Week

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

October 17–21, 2016

Tampa, FL

November 7–11, 2016

San Francisco, CA

software tester certification

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

October 23–25, 2016

Toronto, ON

October 24–26, 2016

Chicago, IL

November 1–3, 2016

Salt Lake City, UT

November 7–9, 2016

San Francisco, CA

November 8–10, 2016

Washington, DC

November 13–15, 2016

Orlando, FL

**November 29–
December 1, 2016**

Atlanta, GA

Phoenix, AZ

mobile application testing

www.sqetraining.com/map

October 19–20, 2016

Tampa, FL

October 23–24, 2016

Toronto, ON

November 9–10, 2016

San Francisco, CA

conferences

STARCANADA

<https://starcanada.techwell.com>

October 23–28, 2016

Toronto, ON

Hyatt Regency Toronto

Agile Dev, Better Software & DevOps East

<https://bsceast.techwell.com>

November 13–18, 2016

Orlando, FL

Hilton Orlando Lake Buena Vista

Mobile Dev + Test and IoT Dev + Test

<https://mobiledevtest.techwell.com>

April 24–28, 2017

San Diego, CA

Westin San Diego

STAREAST

<https://stareast.techwell.com>

May 7–12, 2017

Orlando, FL

Rosen Centre Hotel

Agile Dev, Better Software & DevOps West

<https://bscwest.techwell.com>

June 4–9, 2017

Las Vegas, NV

Caesars Palace



INTELLECTUAL PROPERTY AND SOFTWARE PROTECTION

To protect intellectual property of computer software, software developers need to go to great lengths to prevent misuse. For most of us who have properly purchased apps and complied with software vendors' licensing rules, it can be very frustrating when the app fails to run due to a licensing issue.

This issue's feature article reminds us how software vendors misfire in an attempt to balance protection of their intellectual property with complicated software licensing schemes that frustrate the end-user. In his article "The Evolution of Software Monetization," Michael Zunke presents a detailed analysis of how end-users really feel about software protection methods. And for another perspective, Bob Zeidman gives a fascinating inside look at how difficult it is to determine if your code has been compromised in "What If Someone Steals Your Code?"

If you ever wanted to know what the open source tool Cucumber is all about, read Matt Wynne's article "What Is Cucumber and Why Should I Care?" He shows you a better way to automate customer acceptance tests.

In "The Mindset of the Agile Developer," Gil Broza presents a perspective of what really motivates software developers. Tom Wessel offers a pragmatic approach for teams struggling to achieve consistent results in "Continuous Process Improvement Using Balance and Flow."

DevOps continues to be a very important theme, and if you aren't sure how it fits into your lifecycle workflow, you must read "Use DevOps to Drive Your Agile ALM" by Bob Aiello and Leslie Sachs.

We truly value your feedback. Let us and our authors know what you think of the articles by leaving your comments. I sincerely hope you enjoy this issue! And please use social media to spread the word about *Better Software* magazine.

Ken Whitaker
kwhitaker@techwell.com
Twitter: @Software_Maniac

Publisher
TechWell Corporation

President/CEO
Wayne Middleton
Director of Publishing
Heather Shanholtzer

Editorial
Better Software Editor
Ken Whitaker

Online Editors
Josiah Renaudin
Beth Romanik

Production Coordinator
Donna Handforth

Design
Creative Director
Jamie Borders

Advertising
Sales Consultants
Daryll Paiva
Kim Trott

Production Coordinator
Alex Dinney

Marketing
Marketing Manager
Cristy Bird

Marketing Assistant
Allison Scholz

FOLLOW US



CONTACT US
Editors: editors@bettersoftware.com

Subscriber Services:
info@bettersoftware.com
Phone: 904.278.0524, 888.268.8770
Fax: 904.278.4380
Address:
Better Software magazine
TechWell Corporation
350 Corporate Way, Suite 400
Orange Park, FL 32073

Contributors



Consultant and software engineer **Bob Aiello** specializes in software process improvement, configuration management and DevOps. Bob is an author, the editor of the Agile ALM DevOps Journal (<http://agilealmdevops.com>) based upon his latest book *Agile Application Lifecycle Management – Using DevOps to Drive Process Improvement* and chair of the IEEE P2675 DevOps Standards Working Group. Contact Bob at Bob.Aiello@ieee.org or link with him at <http://linkedin.com/in/BobAiello>.



Gil Broza helps software organizations build and lead engaged, solid, high-performance agile development teams. Gil guides teams and their leaders in creating effective, humane, and responsible work environments so they truly delight their customers and make a positive impact. He works at all organizational levels and coaches clients in both technical and leadership behaviors. Gil authored the books *The Agile Mind-Set* and *The Human Side of Agile*. Contact Gil at gil@3pvantage.com.



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



Leslie Sachs is the coauthor of *Configuration Management Best Practices: Practical Methods that Work in the Real World* (<http://cmbestpractices.com>), a New York state certified school psychologist, and the COO of CM Best Practices Consulting. Leslie has more than twenty years of experience in the psychology field, working in varied clinical and business settings. Reach her at LeslieASachs@gmail.com or link with her <https://linkedin.com/in/lesliesachs>.



Tom Wessel has more than twenty years of software development experience in the banking, healthcare, cable, satellite, and graphics industries. Tom's experience spans the entire end-to-end software development lifecycle with expertise in the areas of program and project management, quality assurance and control, configuration management, knowledge management, release management, development, and technical support. Tom has worked with technology organizations to plan, implement, and train them on agile principles and evolve their agile discipline. Contact Tom at twessel@eliassen.com.



Programming professionally since the late 1990s, **Matt Wynne** got involved in the Cucumber project at its inception in 2007—first as a user, later as a contributor. He published *The Cucumber Book*, co-authored with Cucumber's creator Aslak Hellesey. In 2013, they co-founded Cucumber Limited with Julien Biezemans, the creator of Cucumber-JS. Matt's colleagues elected him CEO of Cucumber Limited in 2016. Reach Matt at matt@cucumber.io.



Bob Zeidman is the president of Zeidman Consulting and Software Analysis and Forensic Engineering Corporation. An entrepreneur and inventor with twenty-two patents to his name, Bob is a member of the Independent Inventors of America. In 2015 he was recognized by the Santa Clara Valley Section of the IEEE with its Outstanding Engineer Award for his pioneering work in the field of software forensics. His expertise has informed rulings in more than 150 high-profile court cases in disputed intellectual property. Reach Bob at Bob@ZeidmanConsulting.com.



Michael Zunke is vice president and chief technology officer for Gemalto's Sentinel Software Monetization Solutions. Michael joined Gemalto through its acquisition of SafeNet in 2015 and previously held senior research, development, and technology roles at Aladdin Knowledge Systems. He holds several patents in software security and is a specialist in application security for licensing, protection against reverse engineering, and automated technologies to integrate security into applications. Contact Michael at michael.zunke@gemalto.com.

Use DevOps to Drive Your Agile ALM

To achieve continuous integration and deployment, agile ALM provides the framework as long as you follow these simple rules.

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

Developing enterprise applications can be a very complex task involving many essential milestones, stakeholders, and dependencies—many of which can be almost impossible to track on a daily basis. Technology professionals who have been in the game for a while will likely reminisce that years ago it was common to use a software development lifecycle (SDLC) that defined all the tasks required to create a system. These steps were often executed in a sequential fashion that came to be known as a waterfall methodology. Although agile and lean development have attained an almost mythical status, the folks performing the day-to-day work may not fully understand the advantage of one methodology over another.

Application lifecycle management (ALM) expands the scope of the traditional waterfall lifecycle to include all stakeholders involved in creating and maintaining complex enterprise systems. This encompasses a much wider scope than the traditional waterfall methodology. Success requires the wisdom found in an agile methodology and the tactics learned from DevOps best practices. This article will show you how to enable collaboration between development and operations to ensure a robust agile ALM that includes information from all essential subject matter experts.

The goal of any comprehensive SDLC is to define and communicate all steps required to get the work done in a timely manner with minimal risk of error. The SDLC should provide transparency and traceability so that each team member understands how his work impacts others, including any dependencies and potential impact to deliverables.

Agile ALM takes a much broader view than the traditional SDLC. It usually involves many more stakeholders and, more importantly, embraces agile principles and practices to enhance developer velocity with improved quality and productivity. The ultimate goal of having a nimble and efficient software development process is to thrill your customers by delivering business value as quickly as makes sense for the end-user. In

practice, this approach requires that your team understand the deliverables and how they can potentially impact the work being performed by the rest of the team. Most technology professionals are learning that it is essential to embrace DevOps best practices such as continuous integration and delivery.

Identify stakeholders: It can be challenging to identify all project stakeholders, let alone keep up with what each team member is accomplishing on a daily basis. For any project, there is always an initial effort to define requirements, often through agile stories and other descriptive documents. Requirements can—and frequently do—change; therefore, it is an industry best practice to use test cases to supplement the requirements document because the testing documentation needs to be kept up to date and may even vary by release.

The best way to understand what each group is doing is to take a collaborative and cross-functional approach. Enhancing communication and teamwork is exactly where DevOps principles and practices are most valuable. With large systems, there are usually many complex dependencies, and DevOps teaches us to take a broad systems view to understand the entire system from end to end. We also need to recognize the value of workflow automation in providing both the transparency

to understand precisely what each team member is doing and the traceability to track tasks and key dependencies as they are completed.

In some cases, this approach can help identify resource constraints that could potentially result in delays or other challenges.

Model your workflow: To be successful, you must be able to model and discuss your workflow with the stakeholders and then use automation to track which tasks are completed and which are still being worked on. It is common to use a scrum meeting to discuss the work being completed and any potential barriers that need to be removed. Most organizations also use a ticketing system to track requests, especially those involving the

You want just enough process to avoid mistakes, while spurning the red tape that motivates people to bypass the process, which can lead to many other problems.

operations team. A workflow automation tool can be used to provide just enough information to get the job done and communicate dependencies to the folks charged with completing required tasks. A common challenge is that workflow automation tools are often difficult to use—and even more difficult to tailor to the group's needs. So make sure that you spend time evaluating and selecting the right tools for your specific effort.

Integrate processes: Integrating operation and development processes is another critical success factor. Your change control processes should be as light as possible but still capable of successfully identifying and helping mitigate any technical risk. It is essential to first identify change requests that can be classified as routine or even preapproved, remove them from the change control meeting's agenda, and then focus the change control effort on discussing what could potentially go wrong (e.g., technical risk) if a particular change is implemented (or even the risk if the change request is not implemented). [1] We find that it is especially effective to integrate agile retrospectives with the operational post mortems often held when an outage has occurred. You want to integrate your incident and problem management processes to act as a feedback loop, while iteratively improving your development process.

As always, you want just enough process to avoid mistakes, while spurning the red tape that motivates people to bypass the process, which can lead to many other problems.

Comply with requirements: Your agile ALM should enable IT controls that ensure you comply with your regulatory and audit requirements while also providing enough feedback to senior management to facilitate IT governance. [2]

Establishing an effective agile ALM requires that you take an agile iterative approach and fully understand agile principles and practices. You always want to build in quality from the very beginning. [3] Iteratively defining your process is usually the most effective way to have just enough guidance to avoid costly mistakes without getting bogged down in verbose, time-consuming ceremony. In the beginning of your development effort, it is usually best to start with the bare minimum of process and then iteratively add controls as you get closer to a delivery date and have a clearer idea of what rules are necessary to avoid costly mistakes.

The two main points to keep in mind are that (1) agile ALM accelerates velocity, and (2) DevOps ensures that you broaden the effort to include all key stakeholders in the most productive and efficient manner possible. **{end}**

CLICK FOR
THIS STORY'S **REFERENCES**

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

Check out the **TechWell Happenings Channel on YouTube**.

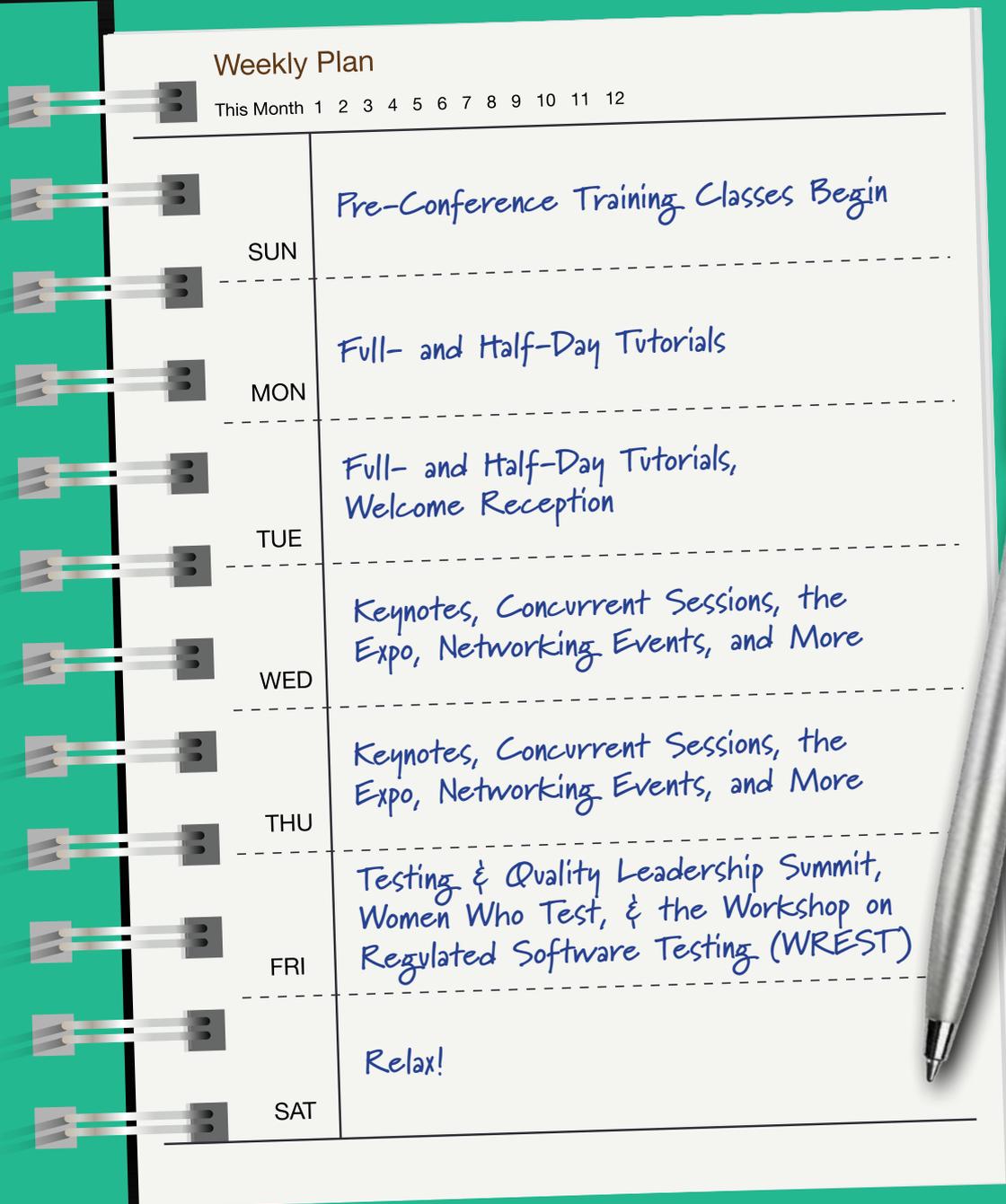
Hundreds of interviews, lightning talks, and STAREAST, STARWEST, and Better Software conference presentations are grouped by topic, so it's simple to take control of your learning experience.

Covering software testing and development topics ranging from mobile testing to enterprise-level agile development and pretty much everything in between, **TechWell Happenings Channel** 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 latest interviews and TechWell conference presentations.

START PLANNING

YOUR WEEK OF LEARNING AT STAREAST



May 7-12, 2017
Orlando, FL

<https://well.tc/SE17MYC>

STAR EAST

A TECHWELL EVENT

How Netflix Embraces Complexity without Sacrificing Speed: An Interview with Casey Rosenthal

Years in Industry: 15

Email: croenthal@netflix.com

Interviewed by: **Josiah Renaudin**

Email: jrenaudin@techwell.com

“At Netflix, we’re very averse to process. We emphasize freedom and responsibility. We don’t have budgets. We only hire senior engineers. That makes it a really dynamic environment that moves very quickly and does things very well.”

“We see chaos engineering as a new discipline within software engineering, designed to surface systemic effects in distributive systems, particularly like ours—where we do have so many subscribers, where we are running at scale.”

“Because we don’t have process, because we actually go out of our way to avoid process, there’s no mechanism for us to even evaluate whether everyone is following agile—let alone recommend or enforce that on engineering teams.”

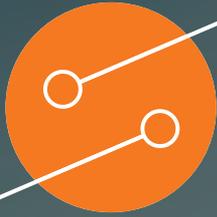
“Even when we have a very popular show release, our control plane is generally scaled well enough to handle a burst in traffic. Because we have so many users across so many regions around the globe, a large burst for us tends to not be as ‘bursty’ as for other businesses.”

“Chaos engineering and intuition engineering, and traffic engineering for that matter, are all services that are designed to help embrace complexity and allow the organizations to still move quickly.”

“Netflix benefits from only hiring senior engineers and really focusing on always raising the bar in our talent. I understand that financially, not every company can do that.”

“While I’m sure that there are some teams here that subscribe to agile methodologies, in a lot of cases, it doesn’t even make sense for some teams to adhere to that.”

“As the systems we work with become more and more like black boxes to us, it becomes more important for quality assurance people and testing engineers to have tools that they can use to describe the properties of the black boxes.”



infostretch

Accelerating the mobility of things



Get to market faster by shifting from QA to QE



Align QA and testing to customer experience and business assurance



Focus on quality right from ideation to commercialization



Prioritize testing with predictive analytics and continuous feedback

Free consultation at infostretch.com/getstarted

The Evolution of Software Monetization



by Michael Zunke

Today's software end-users are increasingly connected and mobile. In addition, they want software delivered as a service to the device of their choice, and they only want to pay for what they use. These changes are fundamentally shifting the way software vendors conduct business.

The adoption of cloud computing, virtualization, advancements in mobile technology, and even day-to-day experiences using the Internet have impacted software end-users' expectations. Whether they're looking to free themselves from rigid licensing restrictions, embrace pay-per-use models, or use self-service entitlement management, the way people want to consume software is rapidly evolving.

Users of all forms of software—from consumer applications and enterprise services to the software in intelligent devices—seek “anytime, anywhere” access, user-centric licensing, and subscription consumption. However, vendors have been slow to adopt new business models that enable them to deliver the experience their customers want. Many of those who have been in the game for a while are still clinging to their on-premise or traditional hardware roots, but as the demands of a new generation of users start to shift, these vendors need to embrace the evolution—or become extinct when users look elsewhere to meet unfulfilled needs.

What Is Software Monetization?

For many software vendors, *monetization* simply means collecting revenue by selling rights to use the intellectual property they've developed into valuable code. For them, monetization challenges revolve around protecting the revenue stream and defending themselves from piracy, theft, reverse-engineering, and misuse of software. But, beyond just protecting revenue, monetization involves enabling and sustaining revenue growth.

Delivering software in ways that customers want to consume it is now a prerequisite in the ultra-competitive software market. Building relationships based on trust is the key to selling anything, and traditional software licensing has become a barrier.

Software vendors need to look at licensing not exclusively as a rights-enforcement hammer, but rather as a way to differentiate themselves from their competitors while providing a rich user experience—or at least one that doesn't leave the customer feeling cheated or confused.

How Do Users Feel about Licensed Software?

To take a closer look at software monetization challenges faced by vendors, as well as software users' disappointments with enterprise software vendors, Gemalto commissioned a survey of six hundred enterprise software users at enterprise organizations with 500 or more employees from the United States, United Kingdom, Austria, France, Germany, Japan, and Switzerland. Close to 200 independent software vendors (ISVs) with at least ten employees also were surveyed. [1]

The survey research revealed that the vast majority of all respondents (85 percent) think software vendors need to con-

stantly adapt to evolving market needs. In addition, 83 percent said that software packaging must be flexible to meet a variety of customer needs, and 81 percent said that software should be accessible across multiple devices. An additional 81 percent agreed that software needs to be future-proof to be successful.

Users Are Frustrated

Of the end-users surveyed, 90 percent said their organization is experiencing challenges with their software licenses. Among the top difficulties cited were inflexible license agreements that don't meet business needs, usage audits, slow time to activation (long customer on-boarding), and lost licensing keys (figure 1).

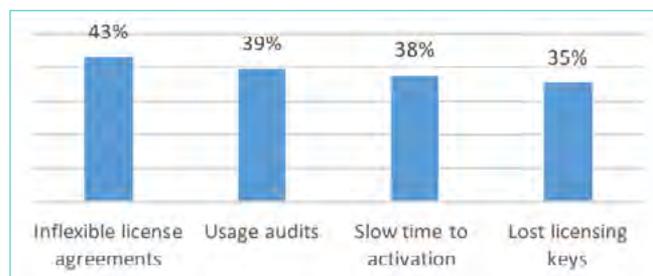


Figure 1: Top licensing challenges (source: Gemalto NV)

When the same six hundred end-users were asked how ISVs could improve their service, most agreed or strongly agreed that clarity around audits, usage tracking, pricing models, and license enablement were all areas for improvement, as shown in figure 2.



Figure 2: Biggest areas for improvement (source: Gemalto NV)

In addition, the survey asked enterprise users their opinions about packaging and delivery options, and separately asked ISVs about the types of packaging and delivery options they're currently providing to their customers. Figure 3 highlights the two main areas of difference: per feature and metered usage, where vendors don't seem to be keeping up with user demand.

Traditional Software Monetization Is Still a Challenge

While end-users are frustrated, software vendors are facing challenges monetizing their software, especially with back-office tasks and licensing enforcement.

Among back-office challenges, 83 percent of ISV respondents claimed frustration with the cost of renewing and managing licenses, 82 percent expressed frustration with the time spent renewing and managing licenses, and 82 percent were frustrated with the research and development time spent on non-product-related development.

In addition, 68 percent of ISV respondents said it was somewhat difficult to get visibility into how their products are being used. A quarter of respondents (24 percent) said it was either very difficult or extremely difficult to get that visibility.

Among ISV organizations surveyed, no more than half are using any single type of software monetization tool to ensure the security, flexibility, and profitability of their offerings.

Licensing Compliance Is Still a Major Concern

While most software vendors worry about the misuse of their software—from theft and piracy to unintentional licensing violations—almost half of enterprise respondents (48 percent) report that their organization has been out of compliance with at least one of their software licenses shown in figure 4.

This same group of respondents estimated that slightly more than a quarter of their software (27 percent) was unlicensed, half of them (47 percent) saying it's because license agreements are inflexible.

When asked how ISVs could improve their services, 80 percent of user respondents think software vendors could provide more clarity about processes and audits, and 72 percent think software vendors could improve usage tracking and audits.

Monetizing the Internet of Things

Gartner estimates that 6.4 billion intelligent devices will be connected in 2016, up 30 percent from 2015. [2] While this growth is not news, software vendors are still struggling to identify how to monetize the Internet of Things (IoT). When Gemalto asked software vendors and enterprise end-users about monetizing the IoT, 69 percent of respondents said they think it could provide their organization with new monetization opportunities. However, 69 percent think there is a lack of clarity about how organizations can monetize the IoT.

Twenty-one percent of respondents to Gemalto's *State of Software Monetization* survey are already exploring IoT monetization opportunities. Fifty percent of them will explore opportunities within the next year, and 80 percent will explore them within three years. According to the survey, companies are being held back from exploring IoT monetization opportunities due to security concerns (48 percent) and lack of expertise (25 percent).

The Only Constant Is Change

The software market is undergoing a fundamental change. On-premise software, software as a service, and intelligent devices are increasingly being deployed and consumed in cloud-connected environments, changing the customer experience and disrupting industries. As software vendors adapt their offerings to meet the evolving needs of the market, several key themes have emerged.

Anywhere, anytime access: Customers expect access to software applications from any device at any time, whether de-

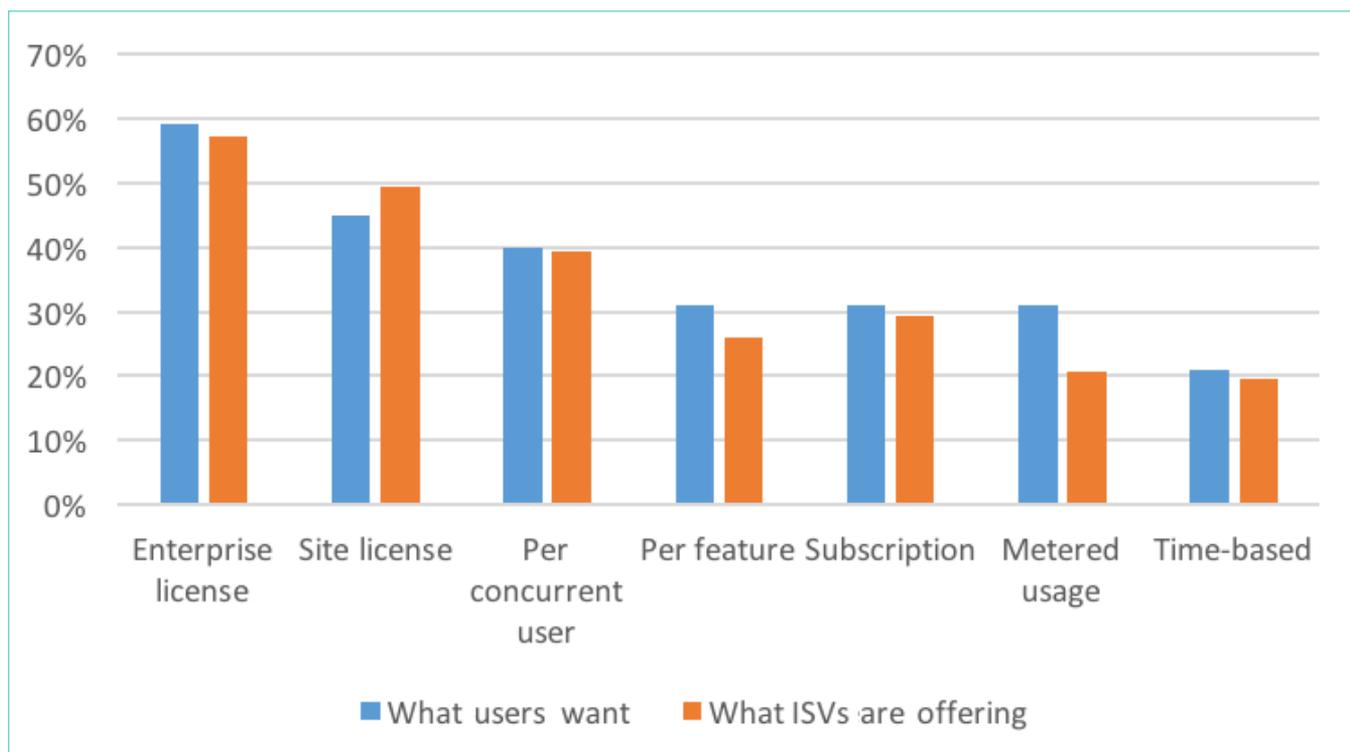


Figure 3: Licensing models (source: Gemalto NV)

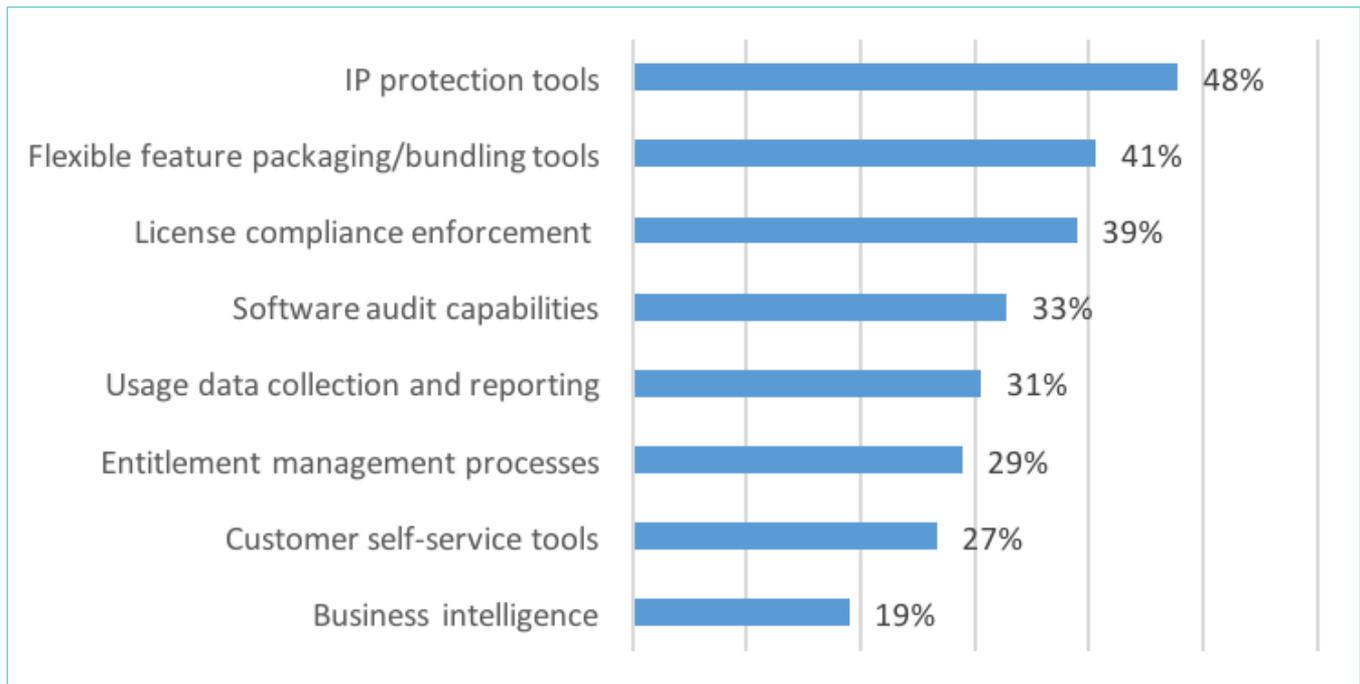


Figure 4: Software monetization tools used by independent software vendors (source: Gemalto NV)

ployed on premise, in the cloud, or across hybrid environments.

User-centric licensing: With end-users looking to access software from any device, licensing mechanisms need to evolve to become user-centric. License delivery and enforcement need to be connected to the individual rather than to the device they're using or the company they work for.

Usage tracking: The growing demand for pay-per-use is driving ISVs to implement tools to track usage to enable usage-based licensing and business analytics.

Common user experience: End-users expect a single licensing experience, regardless of how or where they access software. From the user's perspective, licensing should be consistent across on-premise, cloud, and hybrid environments.

The Future of Software Monetization

It's clear that today's software vendors are not living up to users' expectations. Survey respondents clearly stated that they're out of compliance in part because of inadequate licensing models, and that this is negatively impacting software vendors' monetization initiatives.

Future software monetization solutions need to be able to handle multiple delivery methods and license models with a single back end and enforcement technology. They need to enable granular packaging, incremental revenue mechanisms, and capture usage data—all via the cloud, whether or not the software is cloud-based. These solutions will help software vendors deliver user-centric, à la carte pricing that will provide better experiences and increased monetization.

From an execution perspective, the best licensing solution should have the ability to manage, modify, and update entitlements in real time or near real time via automated services—whether updates are made by the user or the vendor—thus

minimizing engineering involvement. Integration with enterprise resource planning, customer relationship management, and billing systems will be the next level of licensing automation.

Offering flexible license models, tracking and controlling usage, and managing entitlements are common challenges for software vendors providing on-premise software, software as a service, intelligent devices, and even IoT solutions. As software vendors look to adapt their offerings, ISVs need to look toward the future, too. And it is imperative that they make sure the tools and processes they implement today will provide the flexibility to adapt to evolving needs. It's clear that changes currently taking place in the industry are just the beginning. **{end}**

michael.zunke@gemalto.com

[CLICK FOR THIS STORY'S REFERENCES](#)

SOFTWARE TESTER CERTIFICATION

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

Foundation Level Certification

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

Advanced Level Certification

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

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



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



2016 SCHEDULE

Tampa, FL
October 17–19, 2016

Toronto, ON
October 23–25, 2016

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

Salt Lake City, UT
November 1–3, 2016

San Francisco, CA
November 7–9, 2016

Washington, DC
November 8–10, 2016

Orlando, FL
November 13–15, 2016

Tampa, FL
November 14–18, 2016
(Advanced)

Phoenix, AZ
Nov. 29–Dec. 1, 2016

Atlanta, GA
Nov. 29–Dec. 1, 2016

Live Virtual
December 12–16, 2016

SQETRAINING.COM/CERTIFICATION





Development Testing

Prevent Defects Before They Become Features

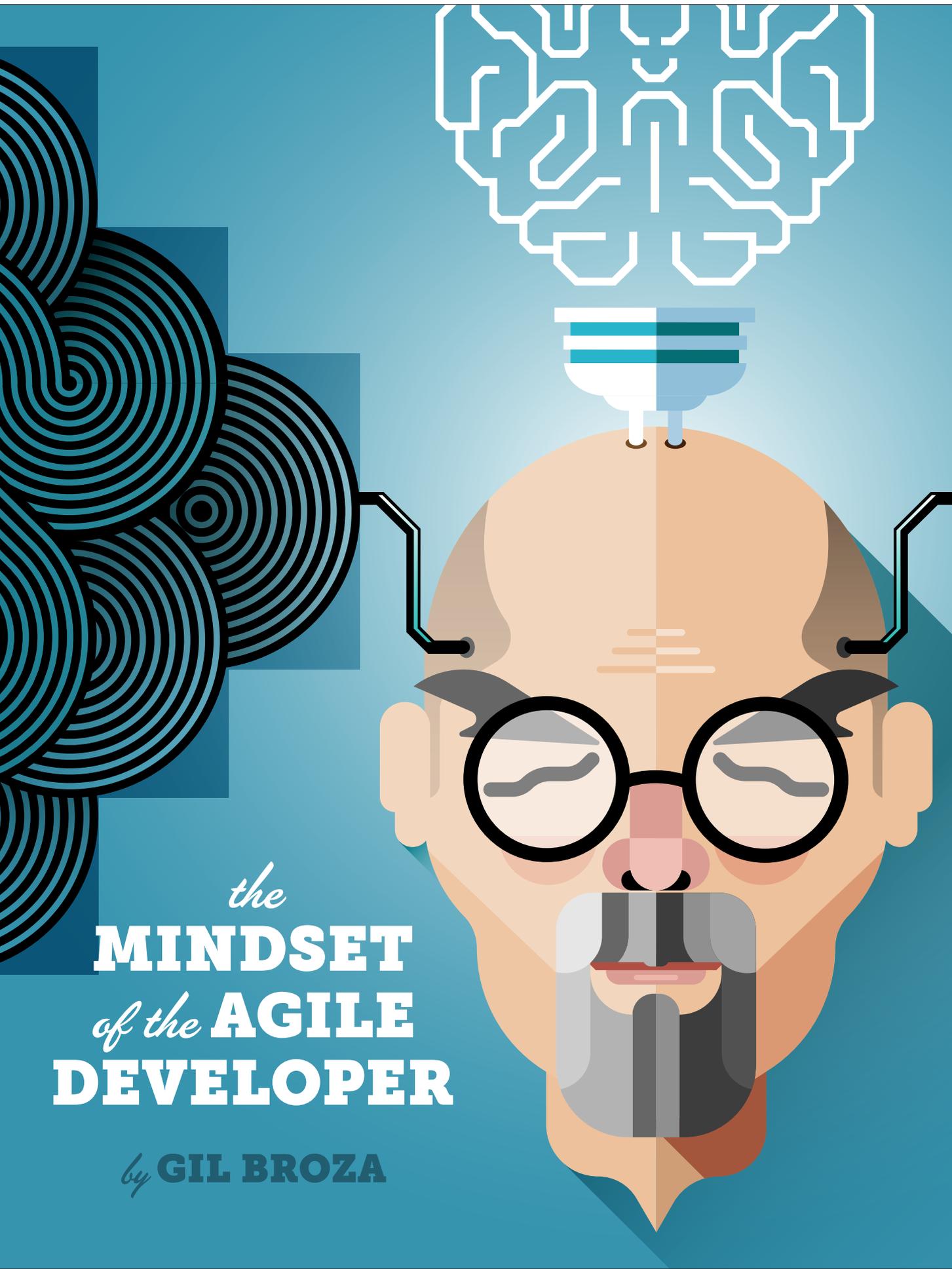
Static Analysis, Unit Testing, Coverage Analysis,
Peer Code Review, Runtime Error Detection

Continuous Testing

Test Features Before they Become Defects

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

www.parasoft.com/bettersoftware2016



the
MINDSET
of the **AGILE**
DEVELOPER

by **GIL BROZA**

Many software developers find themselves thrown into an agile (primarily Scrum) transformation. Everything—practices, artifacts, roles, team structures, and more—changes around them. Not enough developers receive training in applying these moves and mechanics, and far fewer learn how to perform their craft with agility.

To redress the balance, this article explains how agile developers think and act—without attachment to any specific practice, process, or framework.

Doing Agile versus Being Agile

Agile has made serious inroads in software development worldwide. It comes with attractive promises including delighted customers, quality products, and happy teams.

During its twenty years of evolution, agile has somehow developed an incorrect reputation as a methodology, a process, or a set of best practices—so much so that the default mode of adopting agile can be viewed as a recipe.

1. Assemble a small cross-functional team and have them sit together.
2. Designate a business person to be the product owner and a former manager or project manager as the ScrumMaster.
3. Write work items on sticky notes or with an electronic tool.
4. Use planning poker to estimate work.
5. Strive for a continuous integration environment, automated tests, and regular refactoring of code.
6. Demo the completed work to the stakeholders and customer every couple of weeks.

This represents the essence of *doing* agile. There is a false perception that agile practices suffice for accomplishing its promised benefits. However, agile is more than a plug-and-play replacement of methods and steps, akin to the re-

placement of manual labor with electric appliances. Agile is primarily an approach, a way of working, a mindset that guides the creation of results.

Table 1 illustrates the mindset’s significance. Consider the examples of practices, roles, and artifacts associated with agile, their purpose from an agile standpoint, and what they amount to when applied with the previously dominant waterfall mindset.

When you’re *being* agile, you tend to work in an agile-minded manner and can reasonably expect the benefits to accrue. However, going through the motions while holding onto an incompatible approach will not make you more agile.

If you’re seriously interested in the benefits of agile, you must wholeheartedly adopt its mindset.

Introducing the Agile Mindset

One element of a mindset is its principles—standards that guide choices, decisions, and actions. The fundamental principles of agile are to have many short, actionable feedback loops and to continuously learn—both of which feed into continuous improvement of product, process, and teamwork. Agile principles are rooted in the fundamental belief that you do not—and cannot—have all the answers up front.

Agile transforms big work into a stream of smaller work typically by timeboxing or by limiting work in progress. Team results matter more than individual output, and each of the team’s small results has to be shippable, or at least worthy of feedback. It is in service of an outcome—something that advances the project forward, prioritizing being effective over being efficient.

During each small work cycle, the team addresses formerly deferred items whose time has come, experiments with researching and implementing ideas, and simplifies new and previous deliverables.

Practice, role, artifact	Was designed as ...	A waterfall mindset sees it as ...
Daily standup	Frequent check-in to maximize the team’s chance of producing value	Daily status meeting to ensure the team follows the sprint plan
Product backlog	Prioritized list of valuable deliverables the team might get to	Project plan (whether divided into sprints or not)
Pair programming	Collaboration that minimizes the risks of employing humans	Underutilization of expensive resources
ScrumMaster	Servant leader who helps the team succeed together	Project manager in charge of process compliance
Sprint demo	Opportunity to receive stakeholder feedback for increased effectiveness	Frequent deadline for sign-off (and ensuring that people remain busy)
Refactoring	Repayment of interest on technical decisions that allows the business to seize opportunities	Penalty for bad design and not thinking ahead

Table 1: Comparing the agile mindset versus a waterfall mindset

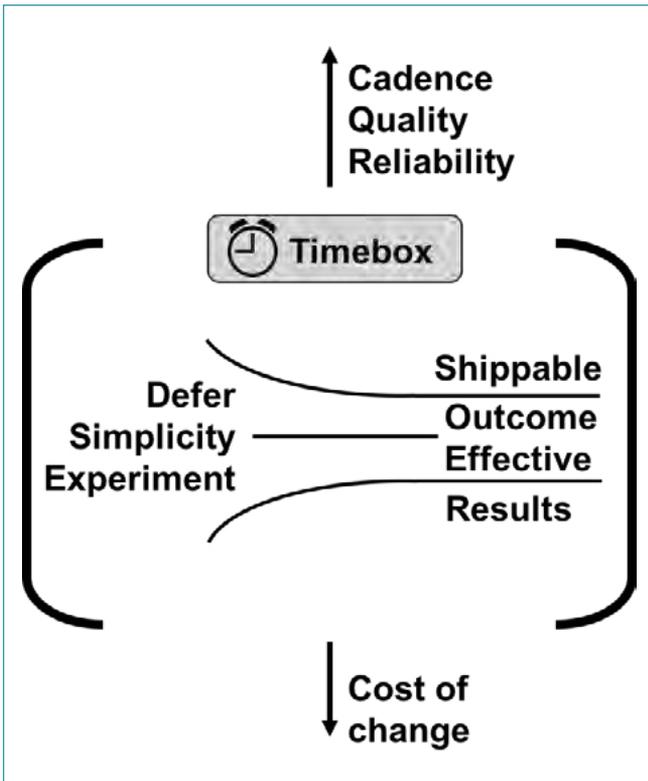


Figure 1: Agile principles regarding work

As figure 1 shows, the team must continuously strive to increase the quality of deliverables and the cadence of product deliveries. Thinking ahead, the team should maintain high reliability in producing results and a low cost of likely changes to accommodate changes of work required to be performed.

Four basic principles allow people to work willingly in a team: respect, trust, personal safety (expecting no harm for acting in the perceived interest of the greater good), and transparency (having easy access to understandable information that underlies decisions). To remain productive long-term, every team member must maximize focus and deliver at a sustainable pace. The team should communicate, self-organize, and collaborate around the work. This assumes the team members share similar approaches in how work is accomplished and that they reach shared decisions using consensus. Facilitate and sustain this human system with a heaping dose of servant leadership, as shown in figure 2.

When the early agile thought leaders joined forces to articulate and name a common approach in 2001, they put down only twelve principles. Still, principles are something you choose, and you might wonder: Why choose these and not others? The answer is that they help you accomplish what you care about—your values. When you employ agile principles, it’s because you care deeply about people, being adaptive, delivering value early and frequently, and continuously collaborating with the customer. My expanded principles work in harmony—and amplify each other—to uphold these values and thereby reach the work’s objectives.

Developing Software with the Agile Mindset

If you’ve worked a certain way for years, applying a different set of principles takes mindfulness, deliberate practice, and constant adjustment. In my experience, most teams new to agile shortcut this learning process by applying practices that others have found to be useful. In doing so, they don’t learn to adopt true technical agility: executing value-adding work in an agile-minded way. Many developers are familiar with Scrum practices for managing work and Extreme Programming practices for writing code. Let’s take a higher level view of technical agility.

Design

Agile reframes and differentiates requirements as problems or needs you have now, ones you *will* have later, and ones that you *might* have later. Believing that the customer prefers to keep options open, the mindset guides you to solve only the problems you have now, without creating a solution so with likely futures that it soon would be thrown out. Repeat this planning exercise frequently, looking for later problems that have materialized as *now* problems. In effect, an agile-developed solution evolves. Employ feedback loops and learning opportunities to inform this evolution, which can be incremental (pieces get added) and iterative (pieces get changed). Design the stable and invariant aspects early and defer design decisions for high-change items to the last responsible moment. Rely on the team’s shared wisdom to determine the best timing.

Always Be Finishing, But Redefine What *Finish* Really Means

Delivering meaningful results early and frequently is one of the agile values. If a desired result takes a lot of work, find smaller meaningful pieces within it. Start with one, get it to *done*, and ship it (or get feedback). In practice, the theory of story splitting requires coding each

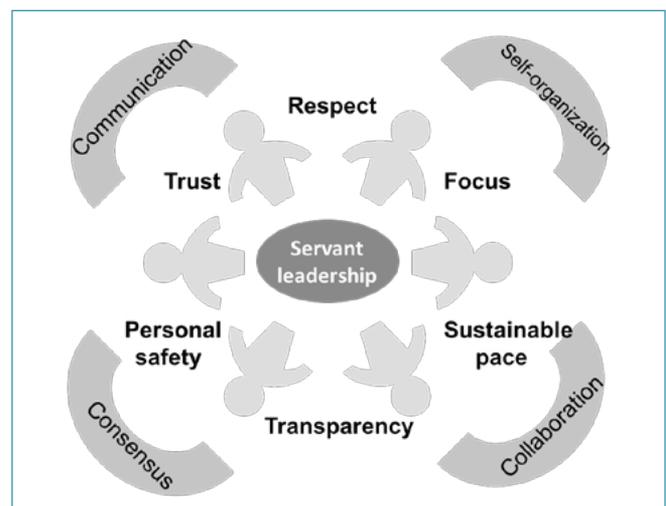


Figure 2: Agile principles regarding people and interactions

subitem (increment) to a high standard of care. Make these increments solid by keeping the design, construction, interfaces, behavior, and flow in lockstep, and don't leave loose ends or empty placeholders. However, developing big work this way might not be as efficient. Its real intent is to make changes easy and cheap. You should constantly think about design, even as you focus your attention on finishing meaningful pieces.

Team Performance

Most companies hire developers who can finish assigned work on their own. From an agile standpoint, a more impactful proposition is what more developers can accomplish through team collaboration. Agile teams outperform their individual members by magnifying individual positives such as creativity, options, knowledge, and experience, and by reducing individual negatives such as fatigue, tunnel vision, misunderstandings, and being stuck. To make this possible, teams cultivate specializing generalists (T-shaped people), avoid silos, sit together, code together, and generally talk a lot. This is another example of teams building resilience to reduce the cost of likely change, rather than minimizing the cost of current work.

Quality from Day One

Even when agile teams have members who specialize in testing, quality is everyone's concern from day one. In addition to keeping the customer delighted, high quality avoids expensive surprises (e.g., integration problems and high-severity defects) and should result in a low cost of change. Teams keep extrinsic quality high through continuous learning and feedback, often manifested by critical thinking and exploratory testing. They also keep the intrinsic quality of complex pieces high through collaboration and simplification. For everything else that tends to be menial, repetitive, and error-prone, use automation.

Safety

Agile practitioners expect to work in a safe organizational environment—to have autonomy within clear boundaries, to act without fear of reprisal, and to retain physical and psychological health. They want to work safely, which to software developers basically means not hearing surprising feedback such as “You just broke this other thing” or “This isn't what I asked for.” When it comes to increasing safety, it's key to decompose large work into small, meaningful elements. By proceeding one element at a time, developers finish each element quickly and get real-time, useful feedback from teammates, tools, and practices such as test-driven development. If that feedback is negative, they can immediately backtrack or undo that task's work.

Code Is for People

If you espouse the agile values of value delivery, collaboration, and adaptability of a team, then you must look beyond optimizing any individual's coding. Instead, ensure the

team's ability to quickly upgrade, change, or fix code. To make that happen, all developers must be able to read and understand any individual's code quickly. In other words, code is for people, not only for computers, which is why agile developers pair up and collaborate to produce intent-revealing, simple, “gotcha”-free code. Their tests maintain the single source of truth, and no design is sacred.

Shifting Your Role and Mindset

Following the agile principles makes sense when the four agile values matter to you. These values are a choice you make to accomplish work objectives, and taking this approach requires stakeholders (colleagues, management, and customers) to make similar choices in order to succeed together.

This is where attempts at agile often hit a wall. Many agile practitioners work with stakeholders and senior managers who have a different value system, one aligned with waterfall values and supported by established methodologies. In that mindset, getting things right the first time matters more than incremental value delivery, making early commitments matters more than adaptation, being on time and on budget matters more than collaborating and co-creating with the customer, and a standardized process matters more than the people who use it.

One reason adopting agile is so difficult is that people's values are invisible, subjective, and rarely expressed. These values are fundamental to being human, and they drive choice—notably, capabilities and behaviors—and, as a result, people don't change values easily. Most companies only change capabilities through training and behaviors through adoption of practices.

Over time, and especially when there's business pressure, they realize that their *approach* to work hasn't changed at all, and their use of agile hasn't resulted in expected business benefits. To adopt the agile mindset, you need to learn new capabilities and behaviors, be mindful of your values and align them with those of your colleagues, and support one another.

Agile is never the destination; it is a means to an end. You can never arrive, but you can get closer. Because agile principles and practices are all chosen in support of the agile values, you can check yourself and your team against those values. Ask yourself: Do you adapt your product, process, and teamwork to change as needed—economically and without drama? Do you deliver meaningful results frequently enough? Do you collaborate effectively with your customers to make mutually beneficial decisions?

If you can genuinely answer each of these questions with a yes, you truly have the mindset of an agile developer. **{end}**

gil@3pvantage.com

SMARTBEAR

Helping Teams Build & Test Quality Software

4M+
Users

25K+
Organizations

194
Countries



TestComplete



QAComplete



TestLeft



Ready! API



CrossBrowserTesting

As the leader in software quality tools for teams, SmartBear supports more than four million software professionals and over 25,000 organizations in 194 countries that use its products to build and deliver the world's greatest applications. With today's applications deploying on mobile, Web, desktop, Internet of Things (IoT) or even embedded computing platforms, the connected nature of these applications through public and private APIs presents a unique set of challenges for developers, testers and operations teams. SmartBear's software quality tools assist with code review, functional and load testing, API readiness as well as performance monitoring of these modern applications. For more information, visit: <http://www.smartbear.com>, or for the SmartBear community, go to: Facebook, Twitter, LinkedIn or Google+.

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)



A photograph of a man standing in the surf at sunset. The man is shirtless and wearing dark shorts, splashing water with his hands. The background is a bright sunset over the ocean with a cloudy sky. The overall mood is serene and active.

*Continuous Process
Improvement Using*

Balance and Flow

by Tom Wessel

Whether you are using an agile or traditional, linear development methodology, success comes down to balance and flow in the delivery of customer value. The challenge is to determine the right balance of people and process to optimize the flow. Finding that right balance in an organization is a challenge that requires the continuous process of inspection and adaption to determine what works—and what doesn't—to generate value efficiently (speed) and effectively (quality). Continuous process improvement should be the mantra of any organization that desires to remain relevant in today's highly competitive and fast-paced technology world.

Software development organizations are complex, adaptive systems that must continue to evolve through inspection and adaptation to not only survive but thrive in an ever-changing business ecosystem. People are at the heart of any organization, and process is the lifeblood that keeps all parts of the organization functioning in harmony. Too much process can clog up the organizational arteries, slowing work to a crawl. Organizations must avoid becoming too slow to deliver value to their customers or too sluggish to react to competition.

However, too little process is equally as dangerous when the various departments of the organization don't function together well—or at all. This can result in communication breakdown, inconsistent quality, and individual heroics rather than an orchestrated effort at production release. The organization may end up getting something to market sooner, but poor quality results in rework, which reduces customer satisfaction.

The discipline of continuous process improvement provides the tools for organizations to find the right balance of process to maximize the flow of value creation.

Lean Change Management

Continuous process improvement involves change, and change can be painful. One might equate this approach to an orthodontist applying braces to a patient. The orthodontist can either use pliers, resulting in extreme pain and suffering but quickly implemented change, or he can use gradual adjustments in tension over time to move the teeth in the desired direction. Most patients would prefer the latter approach—whether it involves pulling teeth or improving the organization.

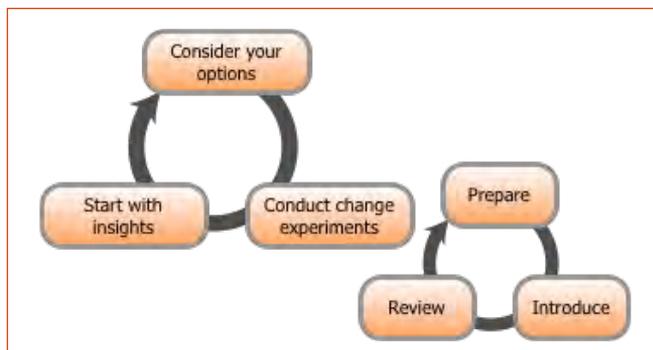


Figure 1: The lean change management process, defined by Jason Little

Developed by Jason Little to introduce change into an organization, lean change management is a lightweight approach that uses a macro feedback loop with the characteristics shown in figure 1. [1]

Insights: The process starts by identifying the current state of the organization via tools such as the ADKAR model, which stands for the five milestones needed to implement change successfully: awareness of the need for change, desire to support the change, knowledge of how to change, ability to demonstrate new skills and behaviors, and reinforcement to make the change stick. [2] The insights are then prioritized based on cost, value, and impact to determine what changes to introduce to the organization and in what order.

Options: Each option will have at least one hypothesis and expected benefits to explore using experiments when the organization is ready to introduce a change.

Experiments: The organization defines an experiment to introduce to the affected people to see if it has the desired effect as shown in the “Conduct change experiments” step in figure 1. Each experiment starts with a preparation step to plan the change and validate it with the affected group. The change is then introduced by working with the people affected by the change. In order to minimize impact and maximize learning, the number of changes introduced in an organization at one time is limited. The cycle ends when the defined time frame to introduce the change is complete, and a review occurs to determine what effect the change has made, what was learned, and how the process can be improved going forward.

Lean change management is an effective, disciplined approach to change management and provides several tools—strategic change canvases, change option canvases, and experiment tracking canvases—to aid your organization in the change process.

Use Kaizen to Identify What Needs to Change

Any discussion on the topic of continuous improvement should start with kaizen. Kaizen is a Japanese word that translates to *good change* or *change for the better* and has its roots in lean manufacturing.

Kaizen was originally introduced by Masaaki Imai in his book *Kaizen: The Key to Japan's Competitive Success*. [3] Today kaizen is recognized worldwide as an important pillar of an organization's long-term competitive strategy by promoting continuous improvement based on certain guiding principles:

- Good processes bring good results
- Go see for yourself to grasp the current situation
- Speak with data and manage by facts
- Take action to contain and correct root causes of problems
- Work as a team
- Kaizen is everybody's business

The spirit of kaizen is that an organization should strive to continually improve in order to remain relevant. Changes, even small ones, move an organization over time to the desired fu-

ture state. Regardless of the size of the change, kaizen means everyone is involved in making improvements, and those improvement efforts—led by senior management as transformational initiatives—can greatly impact the organization.

Kaizen uses a *kaizen event* as a tool to identify and implement meaningful change. These events are small in size, are attended by the owners of a process, and have the ability to improve the process.

Continuous improvement entails introspection and retrospection. While a demo or review in agile focuses on receiving feedback on the product or service being created, agile uses the convention of the retrospective to improve how the team functions effectively to create value. Retrospectives occur on a regular cadence, either following an iteration in the Scrum methodology or at defined times in the continuous flow approach of kanban. The focus of these retrospectives is for the team to identify what is working well and what areas need improvement during the creation process.

The team captures these opportunities for improvement in a backlog and prioritizes them based on the order in which they should be addressed.

Find and Manage System Constraints

Every system has constraints that slow down the flow of value from the customer request to the delivered product. In his seminal book *The Goal*, Eliyahu M. Goldratt introduces his theory of constraints and how using five focusing steps provides a repeatable approach to resolving constraints in a system. [4] The premise is that a system can only move as fast as its slowest part. The steps provide a method to determine the best way to resolve constraints to optimize flow:

- Identify the system's constraint
- Exploit the constraint
- Subordinate everything else to the constraint
- Elevate the constraint
- Identify the next constraint in the system and repeat steps 2, 3, and 4

Let's take the example of a pizzeria. Our pizzeria doesn't produce enough pizza to meet demand during peak hours. Let's first identify the constraint in our system. It appears that the size of the oven is our problem. The oven has a maximum baking capacity of four pizzas. Next, let's make sure we fully exploit the constraint so there are always four pizzas in the oven during peak hours. Once the constraint has been exploited, it makes sense to then subordinate all the preceding steps to limit the amount of unbaked pizza inventory that piles up in front of the oven. Maybe we decide to have fewer associates prepping pizzas because they are overproducing and instead assign some of them to perform other tasks in the pizzeria. Our next step is to determine if we have enough money to invest in a larger oven. The constraint of the pizza oven has been elevated, so we encounter the next constraint in the system.

Each step in the above process provides incremental improvements in throughput, allowing us to meet hungry customer demands in an optimal manner.

Implement Change with the J Curve

Keep in mind that size does matter when it comes to change. Any change effort demands the discipline and planning of effective change management in order to have any chance of success. Process improvement initiatives that take too long to realize benefits risk losing momentum, eventually petering out and dying as team members give up hope. Smaller, successive changes enable quicker wins that maintain momentum, minimize impact to individuals, and increase the chance for lasting change.

Virginia Satir's change process model introduced the concept of the J curve. [5] The idea is that when a change is introduced to improve the current state of performance, there is a period of time when performance actually dips before parity

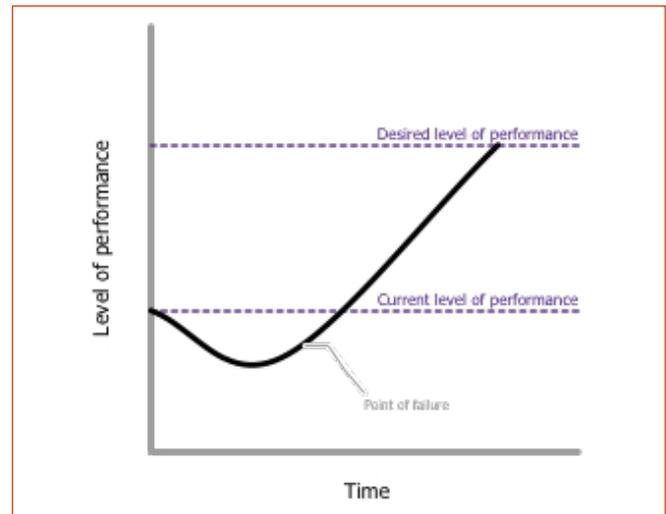


Figure 2: How the J curve tracks overall team performance

is achieved and, eventually, the change improves performance. As shown in figure 2, the longer an organization stays in the bottom of the J—below the current state of performance—the greater the risk that the organization will abandon the change and not realize its benefit.

Smaller changes allow for less time spent at the bottom of the J and for realizing the benefit of the change sooner, allowing the momentum of change to continue. Creating a series of successive J curves allows an organization to incrementally improve performance in small steps to achieve greater change over time.

Use Metrics to Measure Change Results

Metrics are essential to provide an objective measure of whether the change we implemented is having the desired effect. There is a plethora of metrics to consider, but I recommend keeping it simple and starting with just three: speed, quality, and customer happiness.

Starting with speed, it's essential to determine if we are indeed improving the throughput of our system. From the lean world, two metrics that address throughput are lead time and cycle time.

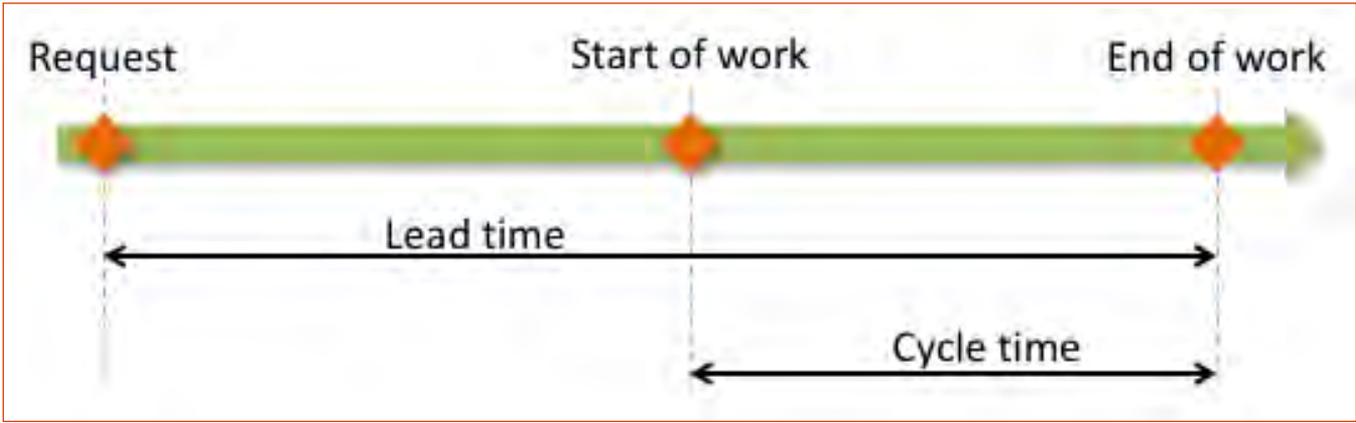


Figure 3: How lead time differs from cycle time

The clock starts for measuring lead time when a customer request is received and stops when the solution is delivered. The clock starts for cycle time—or more specifically total cycle time—when we actually begin working on the request and stops when the solution is delivered. We want to track the average of these two metrics so we can improve team effectiveness over time. Ideally, if our change has the desired effect, lead time and cycle time averages should go down (figure 3).

A key point is that there may be a delay between when a request is received and when it is started. Improving average lead time by shrinking the time between when a request is received and when work is started can be a great area on which to focus.

Just because we're faster doesn't necessarily mean we're better. That's why it's also crucial to measure quality, and a good way to do that is by tracking escaped defects.

To improve the quality of a system and to prevent defects from occurring in the first place, the goal should be to capture the number and type of defects that escape into production. Once an escaped defect has been detected in production, determine the root cause to prevent its reoccurring.

Defect prevention is a quality assurance initiative that saves organizations money as the focus shifts from cost of defect detection and correction to an investment in defect prevention.

So far we have talked about speed (efficiency) and quality (effectiveness). But what about the customers who use what we create? Are we meeting their needs, and are we making them happy?

One metric to consider is your net promoter score (NPS), which determines how likely a user of your product or service is to recommend it to someone else.

Granted, it may be difficult to determine whether the change you made in your process was the sole reason that the needle on customer happiness moved. But NPS is essential and should be monitored on a regular basis.

NPS uses a simple question, rated on a scale from one to ten, to gauge customer satisfaction: "How likely is it that you would recommend a product or service to a friend or colleague?"

Customers who respond nine or ten are considered promoters of your product and will keep buying and using it. Those who respond with seven or eight are considered passives and vulnerable to competition. And those who score six or lower are detractors who are unhappy and can damage your brand.

NPS is calculated by subtracting the percentage of detractors from the percentage of promoters (figure 4).

A healthy customer satisfaction score is in the range of 50-80 percent.

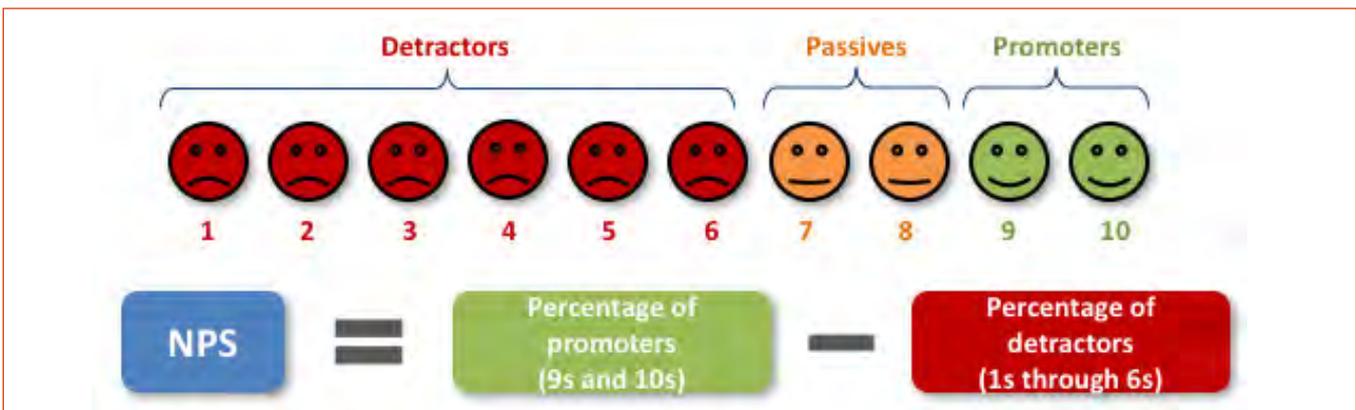


Figure 4: Calculating the net promoter score (NPS)

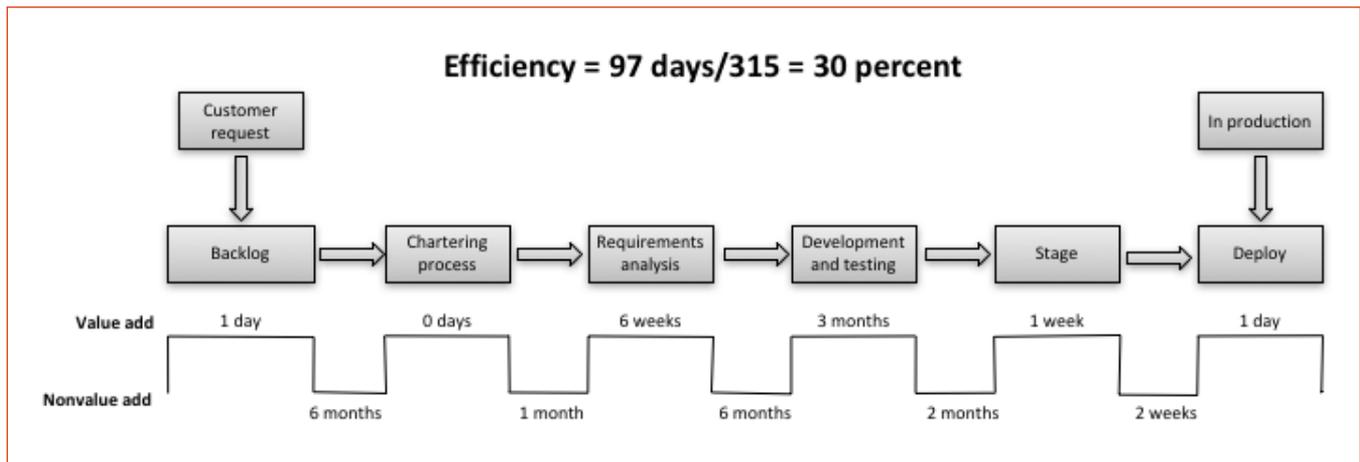


Figure 5: A value stream map showing the key steps in a workflow

Optimize the Whole

Creating a value stream map for each critical process is an essential exercise to improve organizational efficiency. A value stream map that defines key steps in a workflow—from customer request to delivered value—is shown in figure 5.

Start by identifying a key process that is ideal for optimization within your organization. Once the process is identified, determine the seven to ten high-level steps that capture the current state end-to-end process. For each step, determine whether it adds value from a customer perspective. The average time for each value-add and nonvalue-add step is quantified. This allows the organization to determine the total time it takes from customer request to production, how much of that total time adds value, and how much time is spent not adding value.

Once the current state is defined, identify what nonvalue-add time can be reduced as either part of a step or wait state. Then, create a future state value stream map that incorporates the reduced lead time. The organization then can build a roadmap to move from the current to the future state.

Conclusion

Agile and lean techniques have proven to be invaluable in the realm of software development and essential in equipping organizations to embrace change using inspection and adaptation. In any organization the key is to find the right amount of process to optimize the creation of value. {end}

twessel@eliassen.com

CLICK FOR THIS STORY'S REFERENCES

WANTED! A FEW GREAT WRITERS!

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

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

I'm looking forward to hearing from you!

Ken Whitaker

Editor, *Better Software* magazine

kwhitaker@techwell.com



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

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

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

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

IT Services
Business Solutions
Consulting

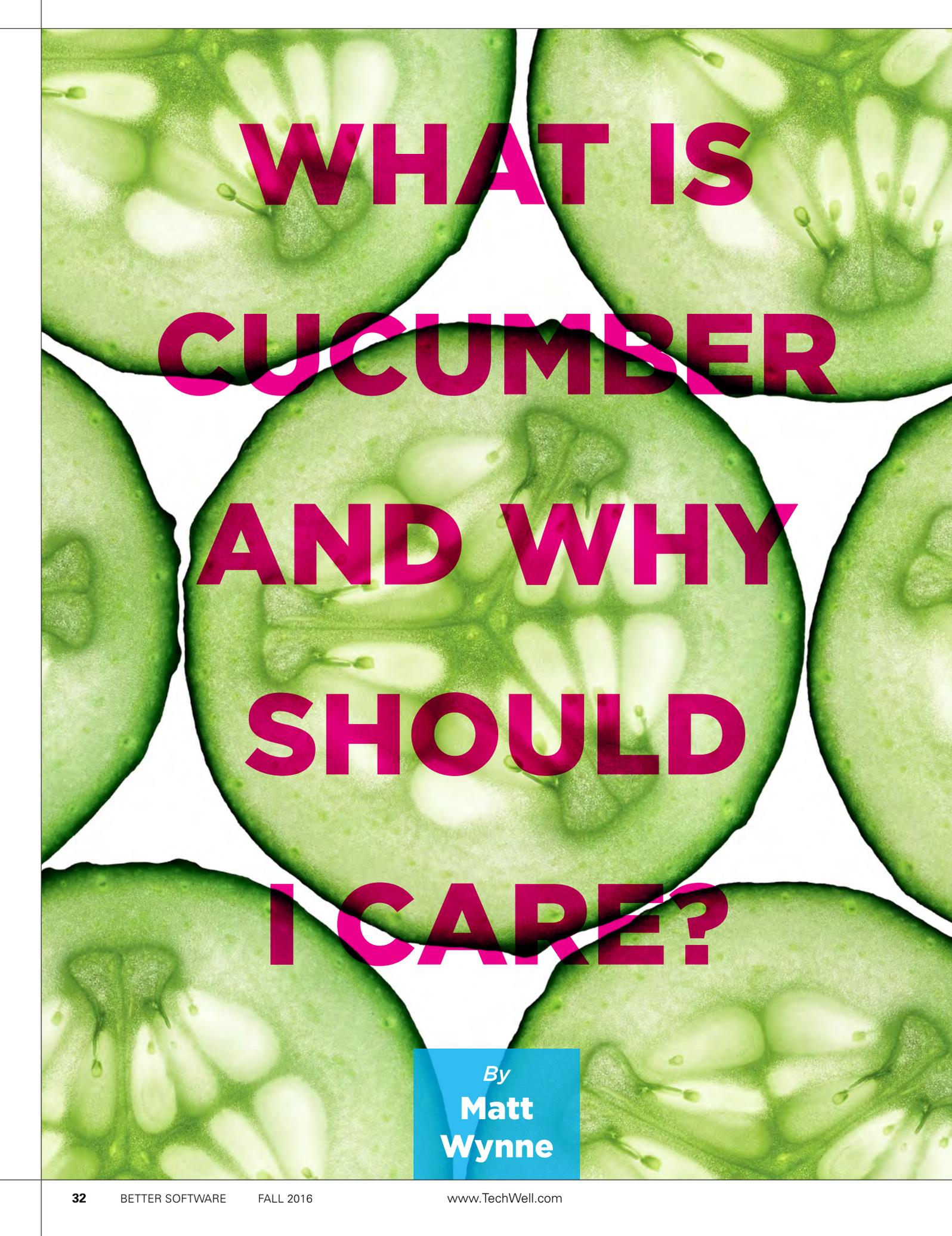


TATA CONSULTANCY SERVICES

Experience certainty.

#ThinkAssurance Today!



The background of the entire page is a close-up, high-resolution photograph of several sliced cucumbers. The slices are arranged in a somewhat overlapping pattern, showing the internal structure of the vegetable, including the green outer skin, the lighter green flesh, and the central cavity containing several small, light-colored seeds. The lighting is bright and even, highlighting the texture and color of the cucumber slices.

WHAT IS CUCUMBER AND WHY SHOULD I CARE?

By
**Matt
Wynne**

Have you heard of Cucumber? Have you used it, or perhaps worked on a team that did? With Cucumber's user base doubling in number every two years, there is a good chance that even if you haven't yet used it, you soon will. In this article you'll learn the basics of Cucumber, its benefits, and how to make the best use of it.

What Is Cucumber?

Cucumber is an open source tool for running automated customer acceptance tests. [1] Before a team starts a development project, the team works together with the customer to agree on concrete examples (called scenarios) that describe the new desired behavior of the system to be developed. A user scenario describing the behavior of automated teller machine (ATM) software might look like figure 1.

Scenario: Customer has insufficient funds
Given Bob has a balance of \$50 in his account And his overdraft limit is \$100 When he tries to withdraw \$200 Then he should be told that he has insufficient funds And he is told that his maximum withdrawal today is \$150

Figure 1: A user scenario

The team can use Cucumber to guide their development, testing each scenario regularly against the system being built.

Figure 2 shows Cucumber's job is to take the examples you've described, run them against your system, and report back whether the system is behaving as expected.

When the tests pass, the code should be ready for the customer. As a result, it can be used to facilitate a process of acceptance test-driven development (ATDD) or, as it is called in some software development communities, behavior-driven development (BDD). [2]

Living Documentation

The previous example illustrates the behavior of a real-world ATM. The real benefit is when a set of scenarios forms an executable specification for the software—a living, breathing document that accurately describes what the system does. [3] You know it's accurate because Cucumber tests it for you every day.

Cucumber's strength is in allowing you to write scenarios in plain English. In fact, Cucumber knows the translation of these keywords in some seventy different spoken languages, allowing you to write your documentation in your business stakeholders' native language.

The only special keywords you need are *feature*, *scenario*, *given*, *when*, and *then*. These keywords are known as Gherkin, the language that Cucumber understands.

The Power of Collaboration

The Agile Manifesto makes a value statement of preferring individuals and interactions over processes and tools. [4] We've all worked on projects where customers and developers failed to effectively communicate about the problem to be solved. It's miserable and results in rework, blame, late nights, missed deadlines, and unhappy customers. When business stakeholders, testers, and programmers work together to try to explain to Cucumber what they want the system to do, something amazing starts to happen.

Cucumber is a tool that facilitates better interactions between individuals. Cucumber requires that a team develop a shared vocabulary for talking about their system. This makes every conversation go more smoothly and creates powerful connections between the business problem domain and the solution domain. Classes, methods, and database tables are given more meaningful names, making them easier to work with in the future.

Artificial barriers between customers, developers, and testers melt away as everyone collaborates to create the best specification they can for the system's behavior.

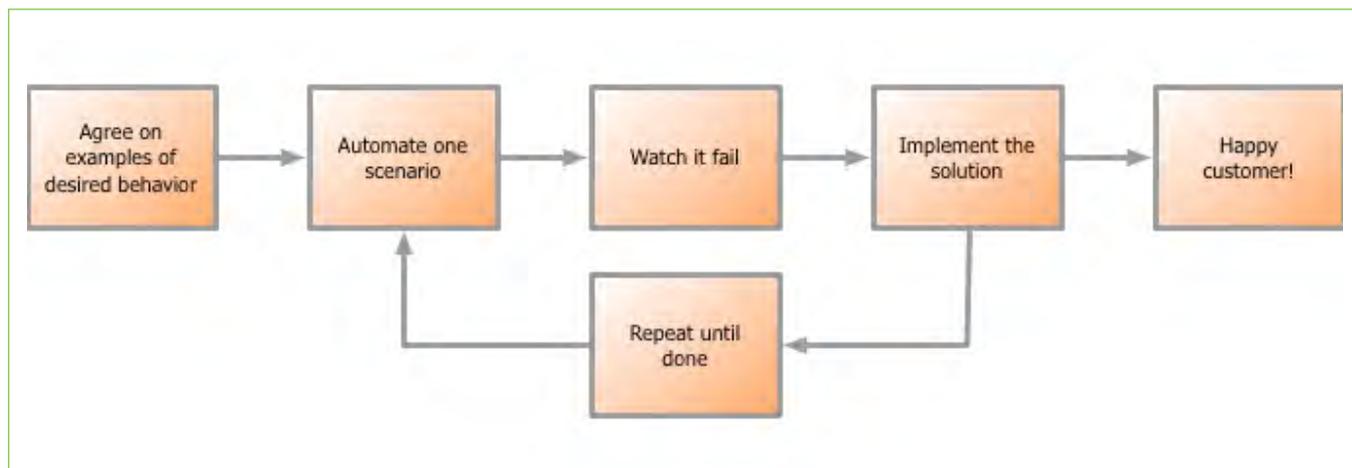


Figure 2: The Cucumber workflow

There's Nothing Boring about Being Predictable

Most Scrum teams have suffered from user stories that were too big to finish in a single sprint, often larger in scope and more complex than originally expected.

Every new Cucumber scenario that is implemented represents a small increment in value of something the software could do today that it couldn't do yesterday.

Working with scenarios helps teams learn the skill of breaking down stories into smaller, yet still valuable, pieces and identifying luxury items (or unnecessary behavior) that can be deferred.

This approach enables the team to focus their energy on implementing the most valuable behavior first and makes sure they're able to meet their commitments to the business in a predictable way.

Flexibility to Automate Anything

Under the hood, Cucumber takes each statement or step in a scenario and looks for an underlying piece of code to run. This code is called a step definition (figure 3).

The team has to supply this library of step definition functions, and it's here that you connect Cucumber to your application so it can run tests against it.

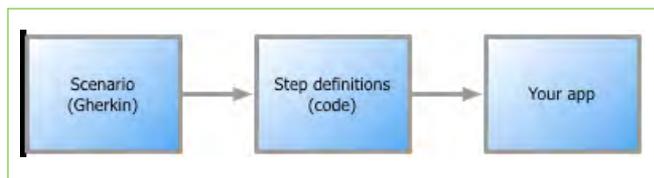


Figure 3: The scenario and step definitions that produce your app

Figure 4 shows an example step definition written in Java.

The `Given` annotation tells Cucumber the plain-language steps that this method will match. When Cucumber sees a step matching that pattern, it will call the `setBalanceForBob` method.

Although initially implemented for the Ruby programming language, Cucumber can run step definitions on almost any modern programming language platform.

This flexibility allows the team to automate their application in whatever programming language they choose.

```
@Given("^Bob has a balance of $(\\d+) in his account$")
public void setBalanceForBob(int balance) {
    app.getAccount("Bob").setBalance(balance);
}
```

Figure 4: Step definition sample code

Embracing Open Source

The world of open source is rich with excellent free libraries for automating almost any kind of application. You can call the user interface directly using tools like Selenium WebDriver for browser-based apps, AutoIT for Windows apps, or Calabash for mobile apps. You can call web service APIs using something like REST Assured or Soap UI, or even make calls directly into your own application code. Many of these ideas are documented in *Cucumber Recipes*. [5]

Back in 2006, I attended a conference where Dan North and Joe Walnes spoke about BDD. I learned about the RSpec open source project, where Dan had been working with David Chelmsky to prototype some of his ideas for bridging the gap between business and technology in what was then known as the RSpec Story Runner. Aslak Hellestøy, who was working in the RSpec team, decided to rewrite Story Runner as a standalone open source project and called it Cucumber. [6]

Through the years, the Cucumber project has grown exponentially with new products—SpecFlow for C#, Behat for PHP, Behave for Python, Cucumber-JVM, Cucumber-JS, and Cucumberish for iOS—as well as the original Cucumber-Ruby, to name a few.

When Cucumber Goes Bad

By now you might be wondering whether Cucumber is the silver bullet, and sadly, it is not. In fact, it isn't unusual for a team to have a bad experience with Cucumber. Every team is different, and every tool or technique has situations where it works well and other situations where it doesn't.

There are several ways Cucumber can fail to deliver the benefits I've described above.

For test automation after the fact: Instead of working in a behavior-driven way where a scenario is automated before application code is written, a team only uses Cucumber to automate tests written *after* the code has been implemented. This code-and-fix method does little to break down the silos between specialisms and never gives the team the opportunity to thoroughly analyze and decompose the problem into bite-sized pieces before they start the work.

Doing it alone: In these projects, test automation code and application code are often written by different people. When testers work alone to add tests to a system, they generally treat it as a black box and default to writing tests

that exercise the entire application stack. This results in large numbers of slow, brittle tests that are costly to maintain.

Ignoring readability: If the team thinks of their Cucumber scenarios as nothing more than tests, they never make much effort to make them work as readable documentation.

That isn't to say these teams are wrong to use Cucumber. For many, this can be their first step on the path to full-on BDD.

How Cucumber Will Benefit Your Project

Adopting Cucumber and BDD as part of your team's toolkit will help you catch defects before they ever get a chance to creep into code. You'll end up with documentation that's always up to date with an entire development team of domain experts.

A Cucumber user with whom I had the pleasure of working summarized the benefits best:

We've found value in the BDD documentation process and obtained a shared understanding of what we're building before it's actually built. Having executable specifications with an automated functional test suite was icing on the cake.

We realized the value of our process when we bypassed it in order to quickly deliver a number of features for a high profile project. We absorbed a new business analyst, QA team, and developers that were unfamiliar with BDD. The result of short-cutting the process was disastrous.

These features had a higher number of defects, did not meet their delivery timelines, had a lack of automated testing, and general hesitation from developers in touching this code in fear of breaking something.

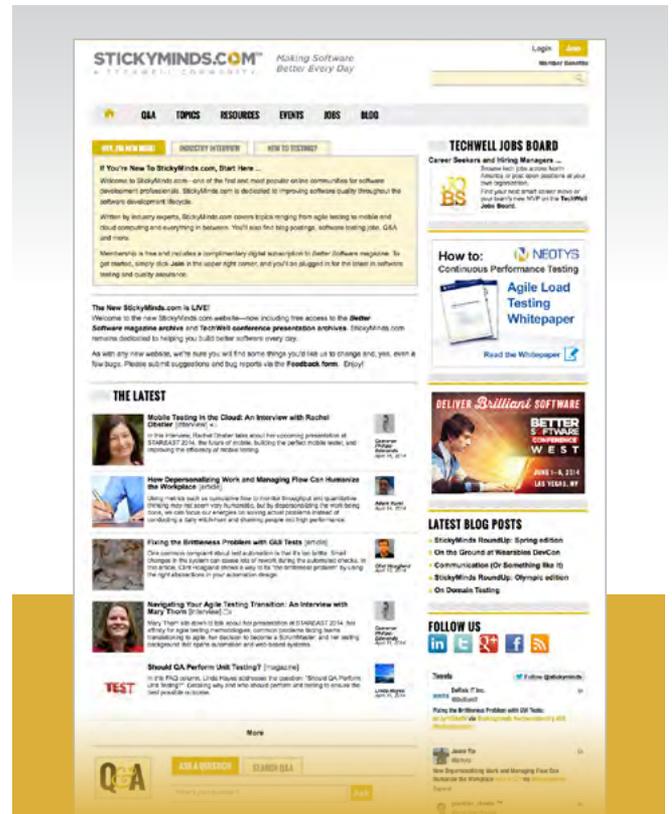
After our experience with these non-BDD implemented features the team (with the support of management) committed to full BDD for all new features.

Learning More about Cucumber

The Cucumber website at cucumber.io provides a great introduction to Cucumber. The site contains comprehensive documentation, a video tutorial series, and resource links to online community forums like the mailing list and chat room. I hope I've managed to share some of that enthusiasm with you and created enough curiosity for you to give Cucumber a try on your next project. **{end}**

matt@cucumber.io

CLICK FOR THIS STORY'S REFERENCES



StickyMinds is one of the first and most popular online communities for software professionals. As a StickyMinds community member, you'll get access to:

- Conference presentations and interviews
- *Better Software* magazine archive—with over 120 issues
- A weekly newsletter and more

Visit <https://well.tc/JoinSM> and join the conversation!



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

Developing and Testing IoT and Embedded Systems: Questions to Ask

By Jon Hagar

Self-driving cars are the new big thing, and they're already being developed by many companies. These "smart" vehicles have sensors, computers, embedded software running on many processors, hardware, and communications through the Internet of Things.

The operational and environment scenarios these cars will encounter are practically infinite. Creating software that is smart enough to deal with the operations of self-driving cars in many different situations will take time, a lot of engineering, and proper testing.

How we should develop and test these systems is a big question, and there are no easy answers.

<https://well.tc/3j6h>

Overcoming the Real Fears behind Behavior-Driven Development

By Kevin Dunne

Behavior-driven development is one of the hottest trends in the testing industry today. It promises greater collaboration, increased automation and test case reuse, and less documentation and waste, just to name a few benefits. It is increasingly seen as the perfect complement to agile delivery and planning processes, and the readability of the tests makes it a great counterpart to DevOps, continuous integration, and continuous delivery, too.

Then why have only 10 percent of respondents to the 10th Annual State of Agile Report said they've tried this technique?

<https://well.tc/3j67>

How to Use the Mobile Revolution to Bolster Your Career

By Josiah Renaudin

From the outside, revolutions in technology can be fascinating to watch. Going from PCs to smartphones and now the Internet of Things, witnessing new pieces of hardware change the way we communicate, artistically express ourselves, and interact with the world range from novel to inspiring.

However, if you're a professional within the industry and fail to take advantage of a sea change like mobile, you're wasting a valuable opportunity. It sometimes feels like decades ago when smartphones came into our lives and turned the world of software on its head, but its growth curve continues to spike, and there are still countless ways you can leverage mobile to inspire fresh career opportunities.

<https://well.tc/3j66>

Scaling Agile: Reasonable Practices for Program Management

By Johanna Rothman

It seems as if everyone is talking about "scaling" agile. What they mean is a strategic collection of projects with one business deliverable: a program.

We don't have "best" practices for agile program management. However, you might find some reasonable practices help you use agile or lean even better.

<https://well.tc/3j68>

Finding a Middle Ground between Exploratory Testing and Total Automation

By Matthew Heusser

Testers seem to be having the same argument over and over again.

The automator wants to get rid of human exploration: to press a button, get a green bar, and ship to production. In some cases, the tool could commit to version control, have something else automatically press the button, and automatically ship to production. This is akin to having robots cut down a forest and stack the wood. No humans involved.

The explorer, on the other hand, wants a human intervention step. They see tools more like a chainsaw. The chainsaw allows the human to go ten times faster, but a human is still in charge, driving the process. The explorer doesn't want robots to do everything automatically; they want to be a cyborg, a six-million-dollar man, to balance the human and the machine.

<https://well.tc/3j62>

Just Enough Testing at Each Stage of Your Delivery Pipeline

By Gene Gotimer

The continuous delivery pipeline is the process of taking new or changed features from developers and getting them deployed into production and delivered quickly to the customer. To reach that goal, the pipeline must be designed to determine whether the software is a viable candidate for production—and if not, then why.

Having frequent quality gates that give feedback about the quality of the software along the way helps us find that answer quicker. Quality gates are decision points along the pipeline where we gather feedback and decide if we want to keep testing. If the software doesn't pass a quality gate, then it isn't worth continuing with that build, and we go back to the developers (or to the requirements) for a fix.

<https://well.tc/3j6u>

Bringing the Value of Your Test Automation Efforts Front and Center

By Michael Sowers

Sometimes, in order to get everyone on board with adopting test automation, you have to build a business case to highlight the potential benefits. Once you've convinced the organization to make that investment, you should determine whether it's actually yielding the predicted benefits—and you'll want to keep these benefits visible to key stakeholders to reinforce the value.

We tend to have many metrics in place to track the progress of testing and measure the degree of readiness or risks in our software products. These metrics include test effectiveness, software quality, test status, resources, issues, and so forth, but what about metrics for the test automation platform? How do we frame the quantitative and qualitative benefits of test automation in a way that links to the organizational objectives and business goals? This is an important element of planning and implementing our automation architecture.

<https://well.tc/3j6b>

Expanded Schedules Pose Project Management Risks, Too

By Payson Hall

A project manager reflected on the challenges that emerged when her customer significantly extended her project's schedule.

The initial goal was aggressive: Get the work done in six months. This would require intense focus by the project team and cooperation from other parts of the enterprise. Soon, it became apparent that competing priorities within the enterprise threatened the project schedule.

The client elected to extend the end date by a year. This “relaxed” schedule was expected to give everyone time to create quality products ... but things immediately began to come apart.

<https://well.tc/3j6a>

Removing the Performance Testing Bottleneck

By Adam Auerbach

When my company started our journey toward continuous testing, our first hurdle was functional testing. We were focused on real-time test automation and ensuring that when code was checked in there were automated tests that validated the functionality. As we matured, we realized that while this was a huge accomplishment, it meant nothing in terms of speed because our performance tests were holding up our production deployments. We needed a new way to think about performance testing.

Around this same time, we were looking at application monitoring tools for production that would provide application performance insights. In doing our proof of concept, we realized that these tools would be a great addition to our non-production testing. We could use them for faster troubleshooting of performance issues.

<https://well.tc/3j6z>

Container-Based Deployments and the Future of IT Operations

By Bob Aiello

Container-based deployments have quickly become the preferred approach for managing the build and release of complex applications. Popular container technologies such as Docker enable developer velocity by providing a robust environment closely resembling production that can be constructed in minutes, yielding flexible sandboxes from just a few keystrokes.

In practice, containers provide well-formed, readily accessible environments, from bare operating systems to databases and other applications. Centralized hubs deliver an easily accessible repository of secure containers that can be quickly downloaded and put into use, accelerating the development effort. Container-based development is both productive and compelling, and it reduces the number of moving parts, which historically was the cause of many mistakes and system challenges.

<https://well.tc/3j6r>

The Merits of a Collaborative Manual and Automation Test Team

By Linda Hayes

In many organizations, the manual and automated test teams are separate. The manual testers have application expertise, and the automation engineers offer scripting skills.

The test team analyzes the test case inventory to select cases that are candidates for being automated. These must be documented explicitly and in enough detail for the engineers to follow without presuming application knowledge. The rest of the cases are tested manually.

<https://well.tc/3j6j>

Work Hard, Play Hard: How Fun Provides Balance

By Naomi Karten

With so much serious stuff going on in the world and at work, there's a role for playfulness in our jobs to serve as a counterbalance. This is not to say that the work shouldn't be treated with all appropriate seriousness, just that a light-hearted attitude might reduce the intensity of the demands and pressures.

By playfulness, I don't mean formal, scheduled timeouts in the workday for everyone to get together and blow bubbles (not that there's anything wrong with that). I'm thinking, instead, of random little things that can lighten the mood and get people laughing.

I asked some colleagues for examples of things they do at work to promote levity.

<https://well.tc/3j69>

What If Someone Steals Your Code?

Taking the steps to protect your software intellectual property is often a low priority. Are you really willing to take that risk?

by **Bob Zeidman** | Bob@ZeidmanConsulting.com

Fifteen years ago when CBS first tapped the seemingly bottomless well that is the *CSI: Crime Scene Investigation* television franchise, no one could have predicted the genre would reach such heights. Today, entire television networks are devoted to “whodunit” stories that are solved with modern-day forensics by a single fiber or a tiny speck of blood. As fascinating as this area is, most of us will hopefully never be involved in a situation where we need this level of sophisticated investigative skills. But we may be involved with cybercrime. In the digital world, the smoking gun is often digitally encoded and can be identified by scanning hard disks, log files, and user accounts for any hint of wrongdoing.

So, what happens when a digital crime is committed, and how would we go about solving it?

Welcome to Software Forensics

Software forensics is the science of analyzing software to determine whether intellectual property theft has occurred. While it has a lower profile in the news than the many cyber security breaches we regularly hear about, it is the centerpiece of lawsuits, trials, and settlements worth countless billions of dollars when companies are in dispute over software patents, copyrights, and trade secrets.

Whether your own software is worth thousands, millions, or billions of dollars, it is worth protecting—especially if it is key to your business. In valuing your software, take into account how much of your business’s sales rely on it, but don’t discount the effort put into its development, debugging, integration, and testing. All of these add to its value. To replicate the code from scratch, someone would need to pay developers to perform all these tasks again.

Perhaps the most well-known example of software forensics comes from the *ConnectU v. Facebook* lawsuit between Facebook and the Winklevoss twins, which was portrayed and made famous in the Academy Award-winning film *The Social Network*. One pivotal deposition scene in the movie shows Mark Zuckerberg (played by Jesse Eisenberg) waving a report

proving that Facebook’s source code was not stolen from the Winklevoss twins. I watched that movie many times with different sets of friends, always silent and under instruction from attorneys on the case not to disclose that I was the consultant who wrote that report. In that investigation, I scientifically compared both sets of source code using software forensics techniques I had developed. If you freeze the movie frame, zoom in on the report, and use sophisticated photo-enhancing, you’ll see my name on that report.

Whether your own software
is worth thousands, millions,
or billions of dollars, it is
worth protecting ...

Ways You Can Protect Your Intellectual Property

Software can encompass three types of intellectual property (IP):

Copyright: The US Copyright Office defines copyright as a form of protection provided by the laws of the United States for *original works of authorship*, including literary, dramatic, musical, archi-

tectural, cartographic, choreographic, pantomimic, pictorial, graphic, sculptural, and audiovisual creations. [1] *Copyright* refers to the body of exclusive rights granted by law to copyright owners to stop others from copying their work without the owner’s permission.

You (or the company that employs you) owns the copyright when the code is written, even if you don’t register it with the copyright office. Many programmers are under the mistaken impression that software is simply ideas and you have freedom of thought. While the latter part is true, software is more than just a thought: It is a specific implementation of that thought. It’s been developed, debugged, and tested at great expense. So the owner can protect it, and unless you’re the owner, you can’t take it with you.

Trade secret: The Uniform Trade Secrets Act defines a trade secret as information, including a formula, pattern, compilation, program, device, method, technique, or process, that derives independent economic value, actual or potential, from not being generally known to or readily ascertainable through appropriate means by other people who might obtain economic

value from its disclosure or use; and is the subject of efforts that are reasonable under the circumstances to maintain its secrecy. [2] In other words, a trade secret is something that helps one business make money and that the business keeps secret, and that people in similar businesses generally don't know about.

Software contains trade secrets. So do customer lists, marketing plans, product definitions, and almost anything that gives your business a competitive advantage. To protect your trade secrets, you must make reasonable efforts to keep them secret. You don't have to lock everything in a vault, but you should inform your employees that your software contains trade secrets and that they shouldn't show the code around to friends to impress them with the great job they've done. As a programmer, you should put a notice in all your files that the code is confidential property of the company. If you end up in court, this will show that you made a reasonable attempt to protect it.



Patent: A patent for an invention is the grant of a property right to the inventor, issued by the United States Patent and Trademark Office. The right conferred by the patent grant is, in the language of the statute, “to exclude others from making, using, offering for sale, or selling the invention throughout the United States or importing the invention into the United States.” [3] Another way of saying this is that the government allows an inventor to stop anyone from producing his or her invention, for a limited time, in return for telling the public exactly how the invention works. In that way, programmers can understand the patented invention, improve on it, and, in some cases, create a better way of performing the same function that does not infringe on the patent.

Software patents are controversial. Some people love them, and some hate them. One thing to remember is that the government enforces them, and your competitors will probably have them. So even if you hate software patents, you should get them as protection. I often tell companies to ask their programmers, “What are the hardest problems you had to solve in this software?” or “What part of the software are you most proud

of?” Those are the first things to patent. If you're in a small company, pick your unique, core algorithm and patent it.

How Software Detects IP Infringement

Through forensic analysis it is possible to scientifically determine copyright infringement and many kinds of trade secret infringement. These two areas lend themselves to source code comparison, and there are software tools that can compare millions of lines of code, line by line, for correlation and infringement. In the past, methods of code comparison to find copying included hashing, statistical analysis, text matching, and tokenization. These tools compared code and output a single measure indicating whether code had been copied. However, this oversimplification wasn't accurate enough to be usable in court.

After years of research, I developed algorithms for multidimensional software correlation that determines which sections of code are similar for multiple different reasons. Now, an expert can use an iterative filtering process to decide whether the correlated code is due to exactly one of the following reasons: third-party code, code generation tools, commonly used names, common algorithms, common programmers, or copying. If the correlation is due to copying and the copying wasn't permitted by the copyright or trade secret owner, then it probably wasn't legal.

The Future of Software Forensics

Today, it's not possible to scientifically determine software patent infringement because software patents cover general implementation rather than specific source code. For example, a program that implements a patented invention could be written in any programming language, using arbitrary function names and variable names, and performing operations in different sequences. There are just so many combinations of ways to implement inventions in software that even the most powerful modern computer technology can't consider every possible combination of code that might infringe a patent. This work is left to human experts using their knowledge and experience, but it's a problem that software forensics experts continue to find ways to automate with clever algorithms and simplifying processes.

How Do I Protect My Code?

You should always register your code with the US Copyright Office. This can be done over the Internet very inexpensively—only costing thirty-five to fifty-five dollars. And don't be worried about disclosing your trade secrets. The registration procedure only requires you to submit twenty pages of printouts of the code, and you can redact any trade secrets in the code. By registering your code shortly after you complete it, you get a government record that you are the owner, which not only provides useful proof in court but also may entitle you to extra damages and reimbursement of legal fees if you win a copyright infringement suit.

You must decide which algorithms in your code should be protected with patents and which should be protected as trade secrets. Patents are disclosed to the public, and twenty years after the filing date of the application, anyone can use your invention. During that period of time, the government rewards you for disclosing it by giving you the right to exclude others from using it. Trade secrets can protect your software indefinitely, but if someone independently discovers your invention, you have no right to stop them. It's best to talk to an IP expert about the tradeoffs before making a decision.

And if you suspect that someone has stolen your code, contact an IP litigation attorney and find a seasoned software forensics expert.

Conclusion

While *CSI: Software* may never get the chance to grace the nation's television screens, it's nevertheless gaining traction among legal professionals and litigation consultants, thanks to the support of advanced software tools. In any event, programmers would be wise to think twice about borrowing a few lines of code for that next project.

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!

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

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

index to advertisers

Agile Dev, Better Software & DevOps East	3
Infostretch	13
Mobile Dev + Test and IoT Dev + Test 2017	4
Parasoft	19
Ranorex	2
SmartBear	24
SQE Training—On-Site Training	25
SQE Training—STF/ADV Certification	18
STAREAST	11
StickyMinds	35
TCS	31

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 © 2016 by TechWell Corporation 350 Corporate Way, Suite 400, Orange Park, FL 32073, 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.