

# BETTER™ SOFTWARE

A TECHWELL™ PUBLICATION

## WATERFALL SPRINTS

Is there trouble in River City?

## SOFTWARE LICENSING

A radical view of handling today's licensing challenges

# Chasing



# Mavericks

USING ROLLING WAVE PLANNING TO  
TRANSFORM AGILE TEAMS



# DEVELOP *and* TEST *for the* MOBILE FUTURE

Mobile Dev + Test Conference 2015 addresses mobile development for iOS and Android, test, performance, design, user experience, smart technology, and security. Hear from experts in the field about where the future of smart and mobile software is headed. Attendees will be able to learn from in-depth tutorials, hear from key experts in the industry, and experience innovative solutions from leading organizations while traveling the Expo floor.



MOBILE  
DEV + TEST

APRIL 13-17, 2015  
SAN DIEGO, CA  
Manchester Grand Hyatt

More details coming soon at  
[MOBILEDEVTEST.TECHWELL.COM](http://MOBILEDEVTEST.TECHWELL.COM)

# Call *for* Speakers

You are invited to submit a speaking proposal for the upcoming spring/summer 2015 conferences based on your experience and insight into software development—the software community wants to hear your story! We want to learn more about your case studies, success stories, innovations, and practical lessons that conference attendees will be able to take back to the office and use in their daily professional actions.



## STAREAST

May 3-8, 2015 • Orlando, FL  
DEADLINE FOR PROPOSALS: **SEPTEMBER 14**



## STARCANADA

June 2015 • Vancouver, Canada  
DEADLINE FOR PROPOSALS: **NOVEMBER 3**

Topics such as test management, testing techniques, test automation, agile testing, cloud testing, mobile testing, and personal excellence are of particular interest to STAR delegates.

## SPEAKER BONUS

Accepted speakers get a complimentary “Conference Only” registration to the conference at which they are selected to speak (a \$1,995 value).



## Agile Development Conference & Better Software Conference West

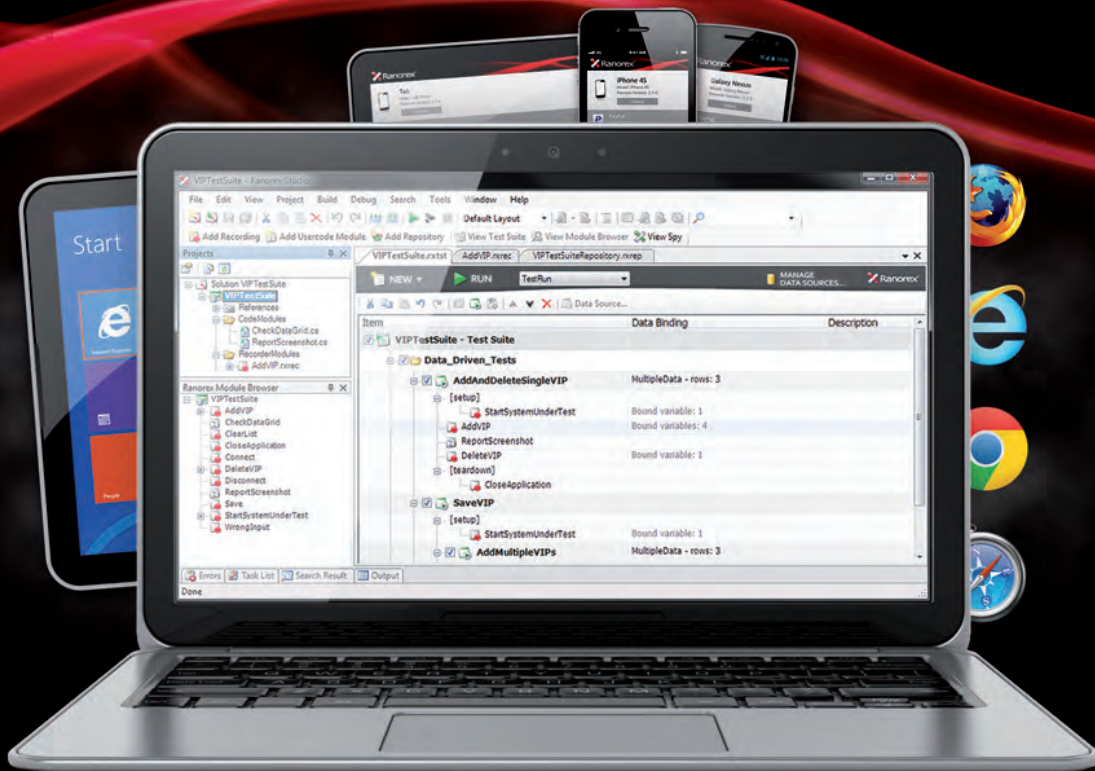
June 7-12, 2015, Las Vegas, NV  
DEADLINE FOR PROPOSALS: **OCTOBER 19**


For Agile Development attendees topics such as agile, scrum, test-driven development, agile software testing, transitioning to agile, and Lean are of particular interest. For Better Software attendees topics such as managing projects & teams, requirements & design, process improvement, testing & QA, development, metrics, security, and mobile are all engaging topics.

**SUBMIT YOUR TALK AT:**  
<http://vlt.me/speaker>



# Automated Testing of Desktop. Web. Mobile.



 Robust Automation

 Broad Acceptance

 Seamless Integration

 Quick ROI

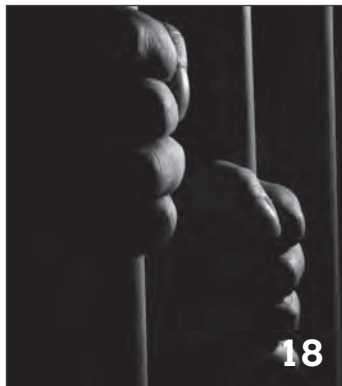


Why Use Ranorex  
[www.ranorex.com/why](http://www.ranorex.com/why)





## CONTENTS



## in every issue

Mark Your Calendar	4
Editor's Note	5
Contributors	6
Interview With an Expert	13
Product Announcements	29
FAQ	30
Ad Index	33

*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](http://BetterSoftware.com) or call 904.278.0524.

## features

- 14 COVER STORY**  
**CHASING MAVERICKS: USING ROLLING WAVE PLANNING TO TRANSFORM AGILE TEAMS**  
By using an approach similar to that used by surfers to catch waves, you can effectively help your team transition to agility. Scott presents a four-stage process that alternates training with coaching and doing with learning.  
*by Scott Frost*
- 18 A RADICAL VIEW OF SOFTWARE LICENSING AND PIRACY**  
Software vendors are making extraordinary efforts to protect the installation and use of apps, but have they gone too far? Preventing software piracy can have an adverse effect on genuine users. Software licensing technology, according to Steve, needs to strike the best balance of protecting the asset while trusting the customer.  
*by Steve Cholerton*
- 22 THE CURIOUS CASE OF WATERFALL SPRINTS**  
It isn't unusual for a project team to believe that adopting a mix of waterfall and Scrum can deliver the best of both worlds. According to Steve, nothing could be further from the truth. In fact, Steve retraces a real project that quickly disintegrated into an absolute disaster.  
*by Steve Zachary*
- 26 MONETIZATION 2.0: THE EVOLUTION OF SOFTWARE LICENSING**  
The cloud and the rapid migration to mobile devices and the Internet of Things have made traditional software licensing schemes obsolete. Omkar describes new software monetization based on business, pricing models, and usage.  
*by Omkar Munipalle*

## columns

- 7 TECHNICALLY SPEAKING**  
**BRINGING QUALITY INTO DEVOPS**  
DevOps is represented by a set of principles and practices that help improve communication and collaboration between development and operations. Bob and Leslie have put together a great introduction showing how quality assurance needs to commence at the very start of a DevOps project.  
*by Bob Aiello and Leslie Sachs*
- 32 THE LAST WORD**  
**LEVERAGING AUTOMATED TESTING TO IMPROVE PRODUCT QUALITY**  
Improving product quality is often a very difficult task for even the best software development organizations. Rajini says the additional benefits of automation include benchmarking, code scanning analysis, end-to-end test cases, and compatibility validation.  
*by Rajini Padmanaban*

# MARK YOUR CALENDAR



## software tester certification

<http://sqetraining.com/certification>

**Foundation Level Certification**  
**September 16–18, 2014**  
Toronto, ON

**September 22–24, 2014**  
Washington, DC

**September 23–25, 2014**  
Austin, TX  
Sacramento, CA

**September 30–October 2, 2014**  
New York, NY  
Portland, OR

**October 7–9, 2014**  
Atlanta, GA

**October 12–14, 2014**  
Anaheim, CA

**October 14–16, 2014**  
Providence, RI  
Minneapolis, MN

**October 20–22, 2014**  
Tampa, FL

## conferences

**STARWEST**  
<http://starwest.techwell.com>  
**October 12–17, 2014**  
Anaheim, CA  
Disneyland Hotel

**Agile Development Conference East**  
<http://adceast.techwell.com>  
**November 9–14, 2014**  
Orlando, FL  
Hilton Orlando Lake Buena Vista

**Better Software Conference East**  
<http://bsceast.techwell.com>  
**November 9–14, 2014**  
Orlando, FL  
Hilton Orlando Lake Buena Vista

**Mobile Dev + Test Conference**  
<http://mobiledevtest.techwell.com>  
**April 13–17, 2015**  
San Diego, CA  
Manchester Grand Hyatt

**October 21–23, 2014**  
Philadelphia, PA  
Salt Lake City, UT

**October 28–30, 2014**  
Raleigh-Durham, NC  
Washington, DC

**Advanced Tester Certification**  
**October 27–31, 2014**  
Washington, DC

## training weeks

<http://sqetraining.com/trainingweek>

**Testing Training Week**  
**September 22–26, 2014**  
Washington, DC

**October 20–24, 2014**  
Tampa, FL

**November 3–7, 2014**  
San Francisco, CA

**STAREAST**  
<http://stareast.techwell.com>  
**May 3–8, 2015**  
Orlando, FL  
Gaylord Palms Resort

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

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

**STARCANADA**  
<http://starcanada.techwell.com>  
**June 2015**  
Vancouver, Canada  
More details coming soon!

# BETTER SOFTWARE™

A TECHWELL PUBLICATION

Publisher  
**Software Quality Engineering Inc.**

President/CEO  
**Wayne Middleton**

Director of Publishing  
**Heather Shanholtzer**

Editorial  
*Better Software* Editor  
**Ken Whitaker**

Online Editors  
**Josiah Renaudin**  
**Beth Romanik**

Production Coordinator  
**Donna Handforth**

Design  
Creative Director  
**Catherine J. Clinger**

Advertising  
Sales Consultants  
**Daryll Paiva**  
**Kim Trott**

Sales Coordinator  
**Alex Dinney**

Marketing  
Marketing Manager  
**Kim Bryant**

Marketing Assistant  
**Kelly Radell**



CONTACT US  
Editors: [editors@bettersoftware.com](mailto:editors@bettersoftware.com)

Subscriber Services:  
[info@bettersoftware.com](mailto:info@bettersoftware.com)

Phone: 904.278.0524, 888.268.8770  
Fax: 904.278.4380

Address:  
*Better Software* magazine  
Software Quality Engineering, Inc.  
340 Corporate Way, Suite 300  
Orange Park, FL 32073



### SOFTWARE PROVIDERS NEED TO TRUST THE CUSTOMER

Thinking back over the last thirty years, there have always been camps proclaiming that one operating system platform is better than another. Was MS-DOS better than CP/M? How did OS/2 lose the battle to Windows? Who is going to win the iOS and Android fight to be the prevailing platform of choice for mobile devices? These trends will continue to change, forcing software vendors to adapt their apps to work on platforms that users demand.

We are all accumulating devices that can be used in different situations: a desktop in the office, a laptop when traveling, a tablet at the coffee shop, and a smartphone anytime, anywhere. Users expect their favorite apps to work on all of these devices. How are software vendors responding to this challenge? Companies like Microsoft, Omni Group, Basecamp, Flowboard, and Adobe have produced innovative solutions that use cloud-based file sharing for storing and retrieving data among apps carefully designed for a wide range of platforms.

As if creating software to run on a wide range of platforms were not difficult enough, enforcing software licensing can become a real challenge. In this issue of *Better Software*, Omkar Munipalle's article on software monetization introduces the changing dynamics of software licenses, and Steve Cholerton's article asserts that the risk of piracy has resulted in software vendors creating restrictive and confusing licensing enforcement. Because many of us rely on software for business and pleasure, the hassle of dealing with restrictive license management techniques appears to be anticustomer. These articles should give you some ideas to streamline how your software products provide licensing protection against unauthorized users without punishing real customers.

Migrating project teams from traditional project management techniques to agile is easier said than done. In his cover article, "Chasing Mavericks," Scott Frost suggests a four-step approach using rolling wave planning to accomplish this. And for those of you working in organizations where management seems to think merging unlike methodologies—like waterfall and Scrum—together can be the best of both worlds, think again. You'll definitely want to consider the lessons learned in Steve Zachary's article, "The Curious Case of Waterfall Sprints."

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 spread the word to your customers and friends about our complimentary subscriptions (<http://www.stickyminds.com/BetterSoftware>). Our subscriber base continues to grow thanks to you!

A handwritten signature in black ink that reads "Ken Whitaker".

Ken Whitaker  
kwhitaker@sqe.com  
Twitter: @Software\_Maniac



## Contributors



**BOB AIELLO** is a consultant, technical editor of CMCrossroads, and the author of *Configuration Management Best Practices: Practical Methods that Work in the Real World*. Bob has served as the vice chair of the IEEE 828 Standards working group (CM Planning) and is a member of the IEEE Software and Systems Engineering Standards Committee (S2ESC) management board. Connect with Bob on LinkedIn at <http://www.linkedin.com/in/BobAiello> or at [bob.aiello@ieee.org](mailto:bob.aiello@ieee.org).



**STEVE CHOLERTON** is a chartered information technology professional, awarded by the British Computer Society, a fellow of the Institution of Analysts and Programmers, a certified ethical hacker, an Oracle Certified Professional, author of several e-books, and a committed and passionate developer of custom and bespoke software for OS X, Windows, Linux, iOS, and Android. Steve designed and wrote the file encryption software used by Leeds Institute of Molecular Medicine, Cancer Research UK. You can reach Steve at [steven@choltech.com](mailto:steven@choltech.com).



**SCOTT FROST** is an agile coach at Davisbase, providing education, leadership, strategy, and change facilitation for its clients, primarily in Houston, Texas. With more than twenty-six years of experience as a technology executive and coach, Scott is passionate about helping individuals and organizations deliver transformational business outcomes with customized coaching and facilitation throughout the enterprise. You can contact Scott at [scott.frost@davisbase.com](mailto:scott.frost@davisbase.com).



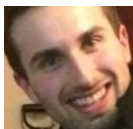
An agile testing coach and process consultant with DragonFire Inc., **JANET GREGORY** is the coauthor of *Agile Testing: A Practical Guide for Testers and Agile Teams* and *More Agile Testing: Learning Journeys for the Whole Team*. She also was a contributor to *97 Things Every Programmer Should Know*. For the past thirteen years, Janet has been working with teams to transition to agile development, and she teaches agile testing courses and tutorials worldwide. Janet contributes to publications and frequently presents at conferences and user group meetings around the world. For more about Janet's work, visit [www.janetgregory.ca](http://www.janetgregory.ca).



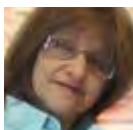
**OMKAR MUNIPALLE** is currently the director of strategy and business development for SafeNet's cloud products. Omkar is responsible for global strategic partnerships, driving new business initiatives, and increased adoption of enterprise software solutions in the cloud. He has more than ten years of experience in the software technology field, focusing on cloud, big data, analytics, and software monetization. You can reach out to Omkar at [omkarnath.munipalle@safenet-inc.com](mailto:omkarnath.munipalle@safenet-inc.com).



As senior director of engagement, **RAJINI PADMANABAN** leads the engagement and relationship management for some of QA InfoTech's largest and most strategic accounts. She has over twelve years of experience, primarily in the software quality assurance space. Rajini actively advocates software quality assurance through evangelistic activities, including blogging on test trends, technologies, and best practices, while providing insights on software testing to analyst firms such as Gartner and IDC. Her official blogs are available at <http://www.qainfotech.com/blog>. She can be reached at [rajini.padmanaban@qainfotech.net](mailto:rajini.padmanaban@qainfotech.net).



A longtime freelancer in the tech industry, **JOSIAH RENAUDIN** is now a web content producer and writer for TechWell, StickyMinds, and *Better Software* magazine. Previously, he wrote for popular video game journalism websites such as GameSpot, IGN, and *Paste Magazine*, where he published reviews, interviews, and long-form features. Josiah has been immersed in games since he was young, but more than anything, he enjoys covering the tech industry at large. Josiah can be reached at [jrenaudin@sqe.com](mailto:jrenaudin@sqe.com).



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



**STEVEN ZACHARY** is an information technology professional with nearly a decade of experience in the design and implementation of software and hardware solutions. As the president of Aleph-One Consulting, Steven has implemented more than a dozen unique high-value projects to clients in the oil and gas, telecommunications, manufacturing, and retail industries. His specialties include Scrum, SAFe, DevOps, computer programming, business analytics, project management, lean principles, metric design, product and release management, test management, and architectural design. Steven enjoys learning about gamification design, machine learning, robotics, and artificial intelligence. Steven can be contacted at [steven.zachary@aleph-one.ca](mailto:steven.zachary@aleph-one.ca).



# Bringing Quality into DevOps

In order to reduce the risk of quality issues and missed deadlines, quality processes need to be a key focus during the entire lifecycle.

by **Bob Aiello and Leslie Sachs** | [Bob.Aiello@ieee.org](mailto:Bob.Aiello@ieee.org) and [LeslieASachs@gmail.com](mailto:LeslieASachs@gmail.com)

DevOps is a set of principles and practices that help improve communication and collaboration between development and operations. DevOps focuses on creating automated procedures to build, package, and deploy applications—often called the deployment pipeline. Organizations that embrace DevOps enjoy development and operations organizations working closely together to share their best practices to ensure that their systems can be updated in a reliable and repeatable way. High-performance teams understand that their quality assurance and testing organizations must be fully engaged in the application build and deployment process. This is precisely where you want to bring quality into DevOps.

DevOps is not just about development and operations. The true spirit of DevOps ensures that subject matter experts collaborate and communicate throughout the entire organization. Building quality in from the beginning requires testing to be performed continuously throughout the entire software or system lifecycle.

Reinforcing that testing and quality assurance take place throughout the entire software and systems lifecycle requires that your team implement testing and quality assurance best practices. Many organizations start with unit testing, which is essential but often limited in scope and test coverage. Functional testing can be helpful, though it is typically restricted to validating those features accessed only through the user interface. There are also much more effective procedures that involve interfacing directly with the application through application programming interfaces (APIs). Programmatically verifying an application can be challenging and requires considerable technical skill and test development effort.

Although quality assurance and testing professionals may be highly skilled in creating robust and effective testing procedures, they rarely have the technical expertise possessed by the developers who are writing and creating code on a daily basis. Developers often get to spend six to nine months learning

a new technology, while the rest of the team may only get a few weeks to get up to speed once the application is ready for testing and deployment. Implementing a robust testing framework requires collaboration between developers, quality assurance, and test engineers. This partnership results in considerable synergy.

Too often, developers find it difficult to test their own code. It is not practical to expect them to switch hats and focus on trying to break their own code. This is where having a strong test engineer can be a huge benefit to the team. Partnering quality assurance and test engineers with their development counterparts ensures that you have both the expertise to understand how the system really works as well as the discipline that is the basis in any effective testing methodology.

As with any specialized skill set, strong testing and development engineers are often in short supply, as are the testing systems themselves. Funding and reserving testing assets can be circumvented by using emerging techniques, including service virtualization testing, to help scale effective continuous testing best practices.

Service virtualization testing allows the test engineers to create virtualized test assets using either a record/playback or programmatic interface. Both of these approaches result in automated testing assets that can be used to simulate testing interfaces for systems that are often in high demand. While effective, these techniques require a great deal of technical expertise, and the best approach is, again, to partner developers with professional quality assurance and testing engineers.

Bringing quality into DevOps requires that you put together subject matter experts who can be most effective at collaborating and developing the testing framework you need. Successful organizations know that getting their best and brightest resources working as a team ensures success. Highly effective teams need to embrace the principles and practices in DevOps that best help maximize quality and productivity. **{end}**

**“Many organizations start with unit testing, which is essential but often limited in scope and test coverage.”**



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

## Live Virtual Courses:

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



## Convenient, Cost Effective Training by Industry Experts

### Live Virtual Package Includes:

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



# SOFTWARE DEVELOPMENT IN THE FAST LANE

**AGILE**  
DEVELOPMENT  
CONFERENCE  
**EAST**

**BETTER**  
SOFTWARE  
CONFERENCE  
**EAST**



REGISTER  
BY OCTOBER 10  
USING PROMO  
CODE **BSME14** AND  
**SAVE UP TO \$400**

GROUPS OF  
**003+**  
SAVE BIG!

**November 9-14, 2014**

**ORLANDO, FL | HILTON ORLANDO LAKE BUENA VISTA**

**Two Conferences in One Location**  
REGISTER AND ATTEND SESSIONS FROM BOTH EVENTS!

Explore the Full Program at  
**ADC-BSC-EAST.TECHWELL.COM**



PMI® members can earn  
PDUs at this event

# Conference Schedule

Build your own conference—multi-day training classes, tutorials, keynotes, conference classes, Summit sessions, and more—packed with information covering the latest technologies, trends, and practices in agile methods and software development.

94 sessions  
offer PMI® PDUs



## SUNDAY

- Software Tester Certification—Foundation Level (3-days)
- Certified ScrumMaster Training (CSM) + PMI-ACP<sup>SM</sup> (3-days)
- Product Owner Certification (2-days)
- Agile Tester Certification—ICAgile (2-day)
- Fundamentals of Agile Certification—ICAgile (2-days)
- Leading SAFe—SAFe Agilist Certification Training (2-days)

## MONDAY–TUESDAY

39 In-depth half- and full-day Tutorials

Multi-day pre-conference training classes continue



## WEDNESDAY–THURSDAY

4 Industry-Leading Keynotes  
32 Agile Development Concurrent Sessions  
32 Better Software Concurrent Sessions  
The Expo  
Networking Events  
...and More!



## Who Should Attend?

- Software managers, directors, CTOs, and CIOs
- Project managers and leads
- Measurement and process improvement specialists
- Requirements and business analysts
- Software architects
- Security engineers
- Test and QA managers
- Developers and engineers
- Technical project leaders
- Testers
- Process improvement staff
- Auditors
- Business managers



## The Expo

November 12–13, 2014

Discover the Top Technologies and Tools All Under One Roof!

**TOOLS • TECHNIQUES • SERVICES • DEMOS • SOLUTIONS**

Looking for answers? Take time to explore the Agile Development Conference & Better Software Conference East Expo, designed to bring you the latest solutions in technologies, software, and tools covering all aspects of software development.

[adc-bsc-east.techwell.com](http://adc-bsc-east.techwell.com)

REGISTER BY USING CODE BSME14 BY OCTOBER 10 AND SAVE UP TO \$400!



# Keynotes

LADIES AND GENTLEMEN...START YOUR ENGINES!

## From Chaos to Order—Leading Today's Software Teams

# 1

**NAME:**  
Ken Whitaker

**COMPANY:**  
Leading Software Maniacs



## The Roots of Agility

# 2

**NAME:**  
Rob Myers

**COMPANY:**  
Agile Institute



## The Future of Agile: Dilution, Calcification, or Evolution?

# 3

**NAME:**  
Jeff "Cheezy" Morgan

**COMPANY:**  
LeanDog



## Get Out of Your Comfort Zone—Now

# 4

**NAME:**  
Tricia Broderick

**COMPANY:**  
Pearson



## Learn SWIFT and Get Ready for iOS 8

*Learn the largest set of changes to iOS and Xcode yet.*

Equip yourself for the in's and out's of SWIFT Apple's new language designed specifically for iOS and Mac OS X. Prepare for iOS 8 the largest set of changes to iOS and Xcode yet.

### FULL-DAY MONDAY

#### A Swift Kickstart: Introducing the Swift Programming Language

Daniel Steinberg

### FULL-DAY TUESDAY

#### iOS 8 Quickstart: The Fundamental Pillars of iOS Development

Daniel Steinberg

### THURSDAY-FRIDAY

#### Agile Leadership Summit

Join your peers and agile industry veterans to explore the unique challenges facing software development leaders as they transform organizations to support agile methods. You'll hear what's working—and not working—for them and have the opportunity to share your experiences and successes

#### THURSDAY:

##### Leading through an Agile Transformation

Mike Kehler, Symantec Corporation

#### FRIDAY:

##### Leadership Evolution in a Hyper-Growth Company

Brian Drummond, LinkedIn, Inc.

##### Empowering Global Teams

Todd Little, IHS

##### Agile beyond Software Development: The Traumatic Brain Injury Program

Ine-Marie Bornman, Henry M Jackson Foundation for the Advancement of Military Medicine (HJF)

##### Think Tank Discussion



**Pollyanna Pixton**  
Program Chair

# DRIVE DOWN COSTS!

## Ways to Save on Your Conference Registration

### Your Best Value—The Full Conference Package (5 Full Days), including:

- 2 Days of Pre-Conference Tutorials
- 2 Days of Concurrent Sessions
- 1 Day of the Agile Leadership Summit
- 4 Industry-Leading Keynotes
- The Expo with over 20 cutting-edge Products & Services
- All Networking & Special Events
- All Continental Breakfasts, Lunches, and Refreshment Breaks
- Combine with the other ways to save below for even more value!

## Exclusive Savings!

Use code BSME14 by October 10 and save up to \$400 off your registration fees (depending on conference package selected).

### Training + Conference

Save \$300 when you combine a training class plus the conference. This package allows you to conserve precious days away from the office and travel time by allowing the intimate learning experience of a training class along with the full conference—all in one location.

## The Larger the Group the More You Save!

See the chart below for an example of how much savings groups of 3+ can enjoy on one of our most popular conference packages—Conference + Two Tutorial Days. To take advantage of this offer, please call the Client Support Group at 888.268.8770 or 904.278.0524 or email [sqeinfo@sqe.com](mailto:sqeinfo@sqe.com), and reference promo code **GRP3**.

Number of Team Members	Regular Pricing	Exclusive Discount (By 10/10/14)*	Group Savings
1-2	\$2,495	\$2,345	
3-9	\$1,996	\$1,876	<b>20%</b>
10-19	\$1,871	\$1,759	<b>25%</b>
20+	\$1,746	\$1,641	<b>30%</b>

\*Full payment must be received by deadline date

Please Note: We will always provide the highest possible discount and allow you to use the two largest discounts that apply to your registration.

#### Premier Sponsor:



#### Platinum Sponsors:



#### Gold Sponsors:



#### Silver Sponsors:



[adc-bsc-east.techwell.com](http://adc-bsc-east.techwell.com)

REGISTER BY USING CODE BSME14 BY OCTOBER 10 AND SAVE UP TO \$400!



## Simon Stewart

Years in Industry: **15**  
 Email: **simons@fb.com**

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

“We’re ruthless about disabling flaky tests and equally ruthless about deleting disabled tests. I think you’ll see other companies doing something similar.”

“Ultimately, the automated tests are taking more and more of the strain out of development because regressions are being caught sooner, and therefore being fixed faster, sometimes before the code has been committed.”

“As I’m sure you’re aware, Facebook has made some spectacular mistakes. So far as I’ve seen, each of them has been taken as an opportunity to understand weaknesses and improve things.”

“One refrain I hear occasionally is that knowing how to program will somehow ‘damage’ a tester, because they understand how the software and machines work. My view is the exact opposite: Knowing how something works gives better insight into potential flaws.”

“I think that there’s a distinction to be drawn between fear and respect. Releasing a new version of the site or the app is rapidly becoming a routine process, and it’s hard to fear something that’s routine. On the other hand, we deeply respect the people who have chosen to spend their time on Facebook.”

“ I absolutely believe that release cycles as compressed as Facebook’s are viable in the long run. In fact, I’d love to see us be able to release even faster—Facebook on the web gets updated twice a day. ”

“I see the trend of ever-quicker releases continuing, and that means that testers will need to be creative about how they do their work.”

“I think that the key thing [for testers] is to want to know how to code. Without that, there’s little point in making the effort.”

**STICKYMINDS.COM™**

For the full interview, visit  
<https://well.tc/IWAE16-5>



A full-page background image of a surfer in a black wetsuit riding a wave. The wave is breaking, creating white foam and spray. The sky is a clear, bright blue. The text 'Chasing Mavericks' is overlaid in a large, orange, 3D-style font with a white dot pattern. The word 'Chasing' is at the top, and 'Mavericks' is below it, both following the curve of the wave. In the bottom right corner, there is white text: 'USING ROLLING WAVE PLANNING TO TRANSFORM AGILE TEAMS BY SCOTT FROST'.

# Chasing Mavericks

USING ROLLING WAVE PLANNING TO  
TRANSFORM AGILE TEAMS  
BY SCOTT FROST



**M**ost organizations approach agile or lean transformations as they approach their entire project portfolio: the traditional waterfall way. In part, these organizations are correct in their assumption that all enterprise initiatives require some up-front planning, alignment, and budgeting. An agile transformation can be a large endeavor, requiring the organization's focus along with constant risk management.

In the movie "Chasing Mavericks," which is about California surfing legends, maverick waves are regarded as somewhat of a mythical legend. These twenty-plus-foot waves, the largest and most treacherous in the world, hold dangerous appeal for surfers—many of whom have died in photo-captured glory. [1]

These massive maverick waves are reminiscent of all the impressively large and risky projects I have seen during my career. With millions of dollars invested in building, upgrading, and replacing business and technology processes in a large single project, these are basically projects that risked the entire company to make game-changing business shifts that rarely went as planned. Success was often judged by whether the initial plan was followed rather than by what was actually delivered. Teams were rewarded when schedules were achieved, even with poor quality, and change requests were kept to a minimum. Too many times, value failures took place.

The rapid industry pace has forced a more radical and successful approach to breaking down large initiatives into smaller, progressively elaborated projects. This approach delivers incremental value along the project continuum rather than gambling on one big rush at the end. Sometimes you will see terms like agile, lean development, rolling wave planning, progressive elaboration, and incremental value delivery to describe these approaches. Adopting these new methods requires changing practices and processes and a culture shift.

There are four proven stages for helping organizations plan their agile transformations. Scheduling is different in every organization, but 90 to 120 days is typical for a product stream of multiple teams. Because transformation initiatives are often viewed and executed as projects themselves, it makes sense that these large transformations should be performed in an agile way—a rolling wave approach rather than trying to “ride the maverick” into agility and hope for the best when the one big wave crests.

## Stage 1: Think Lean about Future States

Glimpsing the future is fundamental to a successful launch, and even more so in transitions that involve not only elaborative planning, but also a serious culture shift of “how we need to work together” principles. There is no substitute for sharing the vision process with the entire team to gain buy-in, common understanding, and commitment.

Lean is about finding ways to free up capacity or costs (waste) within an organization and intentionally reinvesting back into the business; agile seeks to improve quality, increase feedback loops, and adapt quicker to customer change. Both lean and agile seek to improve the way we think about work

and get that work done. While agile follows an evolutionary path by incrementally inspecting and adapting, lean requires a bit more revolutionary transformation, finding the waste and immediately eliminating nonvalue-added work. This balance between agile evolution and lean revolution creates a cadence of transformational rolling waves rather than relying on large, high-risk maverick project waves of change.

Identify the pilot teams by starting with building awareness, then level set product and process expectations through agile vision-planning workshops. Follow this with lean current and future state workshops where all people, processes, and technologies can be visualized in a value stream map exposing problems to be solved and outlining rapid improvements and future states.

The outcomes provide a transformation outline of organizational process and product roadmaps for revolutionary changes, where waste is to be eliminated, and evolutionary waves, where agile teams will deliver products in smaller batches in a continuously sustainable cadence.

## Stage 2: Launch Agile Teams in Rolling Waves

Like surfers on their boards who watch, learn, and wait to take their turns in small groups, agile team launches should take place in rolling waves spaced in a consistent cadence apart from each other. Training should take place initially in an intensive, two or three day boot camp followed immediately with the execution of sprints. Experienced agile coaching and the organization's dynamics can determine how many waves should take place.

Sending some folks from a team off to receive quick training and then expecting them to be transformed on their own while at the same time delivering releases is unrealistic. It is not enough to take a new agile team through early successful adoption. Moreover, teams with a coach avoid missteps and help facilitate complex stakeholder conversations. Nothing takes the place of coaches who have “been there and done that.”

Consider six agile teams desiring two-week sprints with one agile coach. In groups of two-team waves, each team should have sufficient dedicated time with the coach as well as time to provide feedback from which the next group of two teams can learn. Select a four-week cadence between each wave of two-team launches. If things go as planned, all teams should be up and running by week twelve.

If possible, select an experienced coach who is also a great trainer. This will provide continuity when moving from training and planning workshops directly into sprint activities. After about fourteen weeks, all teams should be up and running as effective agile teams (figure 1) with minimal disruption and three waves of learning opportunities to improve your efforts.

To accelerate the success curve, incorporate the recommendations from the stage 1 lean workshops to further refine development tactics. At every cadence interval, gather retrospective insights of what went well and what didn't. Then inspect and adapt.

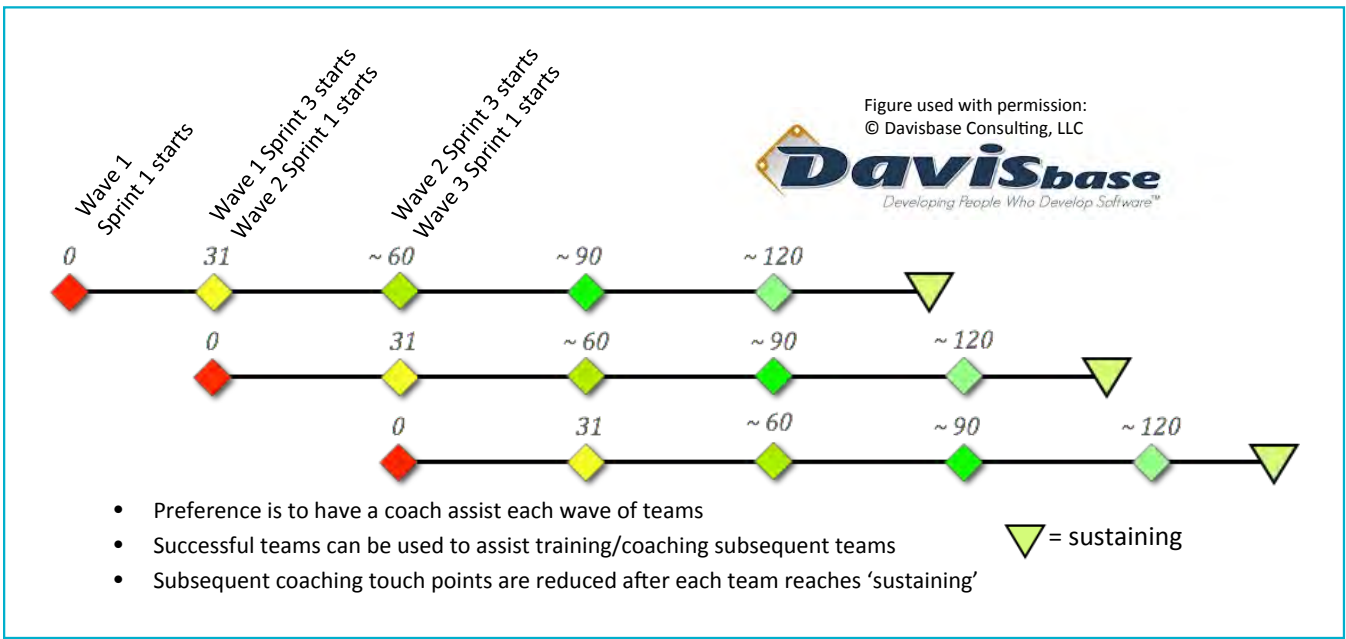


Figure 1: Rolling wave agile team

### Stage 3: Develop Lean, Agile Leaders

Investment decisions, big business ideas, or customer demands are part of the value stream delivery to customers. Early steps like business concepts, budgeting, and program management grow in complexity with time delays. Just as an agile transformation centers around bringing up agile development teams, an agile organizational transformation focuses on those responsible for what the agile development teams work on. This is when it's time for leadership roles to change. Dean Leffingwell's approach, the Scaled Agile Framework (SAFe), adds significant value. [2]

SAFe, well proven in numerous companies, aligns and successfully integrates agile and lean principles into a framework that is easy to understand and implement. Through personal experience, utilizing SAFe feels more natural to organizational leadership than any prior attempts at scaling agile practices outside the development teams. End to end, the entire value stream is addressed in this approach, providing transparency, cadence, and collaboration in a short amount of time through SAFe's three levels: portfolio, program, and team.

While teams become agile teams, overall project leadership implements lean waste removal and improves investment decision quality and planning flow without disrupting the agile teams. As a result, managers are transformed into lean thinking managers. [2] Leaders have a significant amount of work to provide that seamless flow from the product visioning of investments all the way through delivering what customers want.

This holistic approach should positively impact the agile teams who design, build, test, deploy, and support the products and services.

### Stage 4: Create Organizational Clarity and Sustain It

In an agile enterprise, a sustainable pace of continuous improvement never stops. Bringing it all together has some definite benefits:

- The outcome should result in predictable product release cycles that support clear customer and stakeholder expectations.
- Your team should execute at a consistent, sustainable pace (sometimes called flow).
- The organization should transition to both agile and lean objectives by repeating lean events that eliminate waste in the value stream at regular intervals.

The Agile Manifesto has a principle that states:

"Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely." [3]

As an agile community of practice, we know how to sustain a continuous pace. However, the rest of the organization is not yet comfortable on their agile surfboards, so a focus is necessary to build the right habits of sustainability.

Differentiating your product or services in a marketplace requires effective decision making as well as continuous delivery of quality products. Create consistent and committed ceremonies, similar to agile teams, that form these habits, embed a lightweight use of lean techniques, and value feedback and metrics that reward continuous delivery of customer value. If it takes twenty-one days to make a habit, expect this organizational cadence to take up to a year to feel truly comfortable.

Although leadership roles often resist the time and cost commitment for training and coaching activities, it is vital to the successful transformation of the organization.



As lean and agile practices have recurring inspect and adapt components, an enterprise needs to regularly perform retrospectives, especially within the first year. Those activities that add value will become habits you cannot live without, and activities that do not add value will be dropped because you will gain enough experience to know when to let them go.

A best practice you may wish to consider is creating a small center of excellence with a couple of experienced lean and agile coaches to help with the transformation. Given the continuous improvement nature of both agile and lean, a center of excellence can assist an organization with launching new waves or improving existing ones. Outside coaches provide the necessary, objective value that can break through biases during organizational transformations. That being said, enterprises should groom their own internal experts and coaches who can pair with any external assistance and take the lead when they feel ready. The best vendor partners will start with the end in mind—your timely self-sustainability.

## In Summary

Although an organization should be encouraged to chase maverick customer value using agile practices, it should use a rolling wave approach that leverages alternating training with coaching and doing with learning. Use this process:

1. Start with some lean thinking and agile planning.
2. Train, launch, and coach agile teams in small rolling waves, with sustainable and consistent rhythm, incorporating learning with each wave.
3. Develop your leaders into agile enablers and thinkers to feed the development teams in an agile way at a sustainable pace.
4. Create an enterprise that places value delivery above the waste of many of today's decision-making processes.

Gone are the days when an organization could gain a competi-

tive advantage simply by dictating, “go agile.” To gain advantage, visionary organizations are looking for end-to-end value delivery transformation. {end}

scott.frost@davisbase.com

Sticky  
Notes

Click here to read more at StickyMinds.com.

■ References



**Ready to take your software testing career to the next level? ISTQB Certification will lead the way.**

You've proven you have basic software testing knowledge with your ISTQB Foundation Level Certification. Now is the time to take your certification and career to the next level. ISTQB Certification has an advanced career path for every tester. Choose 1, 2 or all 3 ISTQB Advanced Level Certifications:

- ▶ Test Manager
- ▶ Test Analyst
- ▶ Technical Test Analyst

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

**See the ISTQB Software Tester Roadmap and choose your path to greater career success:**

**[www.astqb.org/advanced](http://www.astqb.org/advanced)**



**A RADICAL VIEW  
OF  
SOFTWARE LICENSING AND PIRACY**  
BY STEVE CHOLERTON



**T**here's someone or some company out there who invested their time and expertise building your favorite piece of software. With the exception of free software or open source software, the software creators are certainly entitled to be rewarded for their efforts. In fact, they rightfully deserve it.

If a fee is attached to the use of the software and a user avoids paying the fee, then that is software piracy. Some would argue that it amounts to nothing less than theft. If ten fee-based, physical products are acquired by nonpaying users, then the seller is without those ten items they could have sold to anyone else. That is theft. If, however, you download and use ten copies of software without paying, then you could argue that the seller (or the creator of the software) still owns the property and probably isn't directly and immediately affected by the loss. Because there is no physical product to deal with, software can easily be perceived as a bottomless inventory of digital technology. Would the user have bought the software if he couldn't have obtained an illegal copy? Maybe so, but maybe not.

That is how software piracy differs from traditional theft. What a software pirate has done is taken away the seller's chance of receiving income from the pirate for that software sometime in the future. It's definitely a different kind of theft. The impact of software piracy just isn't clear-cut either way, with laws, policies, and attitudes being firmly rooted in the pre-digital viewpoints of the past.

Assuming that there is a cost to creating, producing, marketing, and supporting software, it stands to reason that there needs to be a mechanism in place that makes it possible for the seller to make a return on the investment. That mechanism is generally known as software licensing.

Preventing software piracy can have an adverse effect on genuine users. Some licensing schemes are so restrictive or complex that people have downloaded cracked copies of the software to use instead, just to save time and effort. In effect, this is like punishing them for their honesty. Whatever form of licensing you use, never punish the genuine, fee-paying customer for the actions of the software pirate.

As a software developer or software publisher, that's not an experience you want for your customers. Of course, whatever software licensing mechanism you decide to use, there will always be some who resent it.

## Striking the Best Balance

The right way to approach licensing is to make it a win-win for everyone.

If, as a developer, you have done your best to minimize the impact to the honest customer while making some effort to thwart the software pirate, then you have done all you can, and any customer who is going to kick up a fuss about your licensing mechanism is unlikely to be a customer you actually want. Most customers appreciate that your company needs to stay in business and realize that to do that, it is necessary to pay for the work you perform.

So, what is a fair software license, for both the supplier and

the customer? What do you expect your customers to do to license their copies of your product? What do you need to do to help protect your product, your sales, your livelihood, and development in a product that is important to you?

It is fair to expect the customer to do something to help the software developers protect their products. After all, if the software developer doesn't stay in business, the software doesn't have much of a future, and the customer risks being left high and dry.

The emphasis should be on the developers to provide a method of licensing their software that can be validated quickly and easily, thus encouraging the customer to purchase and use a legitimate copy of the software rather than paying a visit to an illegitimate download site.

## Why Can't I Purchase One License to Use on Multiple Systems?

If a user purchases software for his own use, then shouldn't he be entitled to copy it onto other computers for use whenever and wherever he likes? The licensed user also should be able to copy the software onto a USB drive, connect the drive to a friend's computer, and use the product there. Why not?

Under no circumstances, however, should a copy of the software remain on the friend's computer, thereby giving her the ability to use it in the licensed user's absence.

This seems to be fair for both the customer and the software developer.

## Trust, Trust, Trust

If software developers don't believe people are fundamentally honest or that they would pay a reasonable price for a quality product, the software industry has in some ways become its own worst enemy. Some software companies have adopted a spider web of nontrusting tactics that are complex, unworkable, illogical, and unfair.

On the other hand, "We need the customers more than they need us" may be a better, long-term perspective that directly impacts how software license protection is designed and implemented.

As a software developer myself, I felt it important to look deeper into software licensing. Every new product is an opportunity to improve the end-to-end purchasing and licensing experience for your customer. Software providers only really get one chance to get it right.

## Designing Licensing Protection to Support How the Software Is Actually Used

Software developers are getting increased pressure to create software apps that run natively on a broad range of platforms: Windows, OS X, and Linux. As the world becomes smaller and smaller, modern software needs to support multiple languages. The software licensing methodology should support all of these uses. Besides, as long as one licensed user has purchased the right to use the software, it can typically only be run on one computer system at a time anyway.

Utility software should stay clear of using configuration files

or registry entries, as that goes against the design of software that may need to be used on a different system. Utility software could easily be viewed as the equivalent of a tradesman's tool in a tool belt.

Can you imagine a screwdriver that is only licensed to be used on a specific object? What if you had to pay for a license for every object on which you want to use your screwdriver? Forget the expense; it's just not convenient or practical. In the same way, software publishers shouldn't punish legitimate users of their software in a vain attempt to stop people from stealing it.

In many respects, how software is protected is usually buttoned up with the software's license agreement (SLA). When the service-level agreement is complex or one-sided, prospects may never become actual customers. Why not make the SLA simple to understand and simple to enforce?

"You can copy and use [product] on your own personal computers, with one instance of [product] allowed to be in use at any one time. You can place [product] on a USB device and plug that into anyone's computer and use [product] from the USB device for as long as you need. All licenses are valid for all supported operating systems, all supported languages, and all supported databases."

This SLA presents an honest explanation as to how the software should be used. It is designed to trust the user and not assume the user is a pirate or include unnecessary restrictions. If the simplicity of the SLA fails, there are several other incentives to help the user make the right decision and become a licensed customer.

Software products also could appeal to customers' financial interest to get their friends and associates to purchase their own software licenses with a referral fee that acts as an active incentive:

"A payment is made to an existing customer thirty days

after a new customer makes a purchase based on an existing customer's referral and quoting the existing customer's referral ID within forty-eight hours of purchase. The payment amount is 10 percent of what the new customer spends (before taxes), made via PayPal."

There are several passive incentive techniques that can also help:

- The licensed user name is displayed in the title bar of the startup window as well as on the main window. This would be a deterrent to piracy, as the original purchaser does not want his name showing up elsewhere and the new user doesn't want to see someone else's name where hers should be.
- Make sure the software is fairly priced. Customers do not want to feel they are being taken advantage of.
- Offer a variety of license types that not only give the user choices but also offer additional discounts for worthy establishments, education, or charities.

Unfortunately, the different types of software licensing that are necessary to give customers the choices they want can bring additional and unwanted complexity.

To keep things simple, the additional license types can all be calculated as  $N * x$ , where  $N$  is the single user standard price, here shown as \$30, and  $x$  is the multiplier. An example is shown in table 1.

This makes it simple for a customer to determine quickly the cost to license. Removing any unnecessary complications and offering transparent pricing make the customer experience more pleasant and empower the customer with the information to make a swift and accurate purchasing decision.

Additional licenses or license upgrades are more likely to be purchased as needed when the customer has confidence in the honesty and transparency of the pricing and appreciates the effort being made to simplify the purchasing experience.

Product	Edition	License Qty	Price (N)	Actual Price
	Single User Standard	1	N	\$30
	Single User Lifetime	1	$N * 3.5$	\$105
	10 User	10	$N * 8$	\$240
	10 User Lifetime	10	$N * 8 * 3.5$	\$840
	Educational Single User	1	$N * 0.5$	\$15
	Educational	Campus	$N * 50$	\$1500
	Site	One Address	$N * 100$	\$3000
	Corporate Purchase	Unlimited	$N * 1000$	\$30000
	Corporate Rental (Monthly)	Unlimited	$N * 20$	\$600

Table 1: Product pricing matrix



The above incentives are designed to cater to potential customers who have obtained illegitimate licenses with the goal of converting them into licensed users.

## Making the Trial Mode Work for Both the Customer and the Supplier

What about the customers who have downloaded trial editions of the software? The software developer needs to give them the chance to see the capabilities of the software, while also convincing them to reach for their wallets.

Trial mode restrictions can make or break a software product's acceptance. Two mechanisms work extremely well for trial usage:

- Restrict some key features (perhaps the maximum number of connections or file size)
- Generate a nag message after specific events or intervals

You should not, however, limit the software's app to a period of time (such as a thirty-day trial) or disable functionality in any way.

The potential customer can still see and access the full capabilities of the product. The trick is to make it less convenient than if they had purchased a legitimate license.

In order for the trial model to be successful for the software developer and the potential customer, it is necessary to again pay attention to the overall customer experience. Out of the box, include a configured sample database (or whatever data the software uses) so the user can experiment with the product immediately.

Trials should include a complete, quality manual. Will all prospective customers read it? Nope. Will a high percentage of folks notice if it's not present? Probably, especially when attempting to use more complex features.

The customer experience is what determines if a sale is made. With so many software products available in online stores, it is difficult enough to get your product noticed in the first place. By trusting the customer and presenting an exemplary customer experience, you've made a customer for life.

{end}

steven@choltech.com

## WANTED! A FEW GREAT WRITERS!

*Better Software* magazine is always looking for authors interested in getting their thoughts published in 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
- Configuration management

I'm looking forward to hearing from you!

Ken Whitaker

Editor, *Better Software* magazine

kwhitaker@sqe.com





# **THE CURIOUS CASE OF WATERFALL SPRINTS**

**By Steve Zachary**



It was a typical, quiet Monday morning at the office. My role was to lead the project management and business analyst functions in a multimillion dollar project focused on modernizing the company's key financial systems. The project had just received funding and resources were being assembled.

As I passed through the hallways toward my desk, the executive who led the IT group signaled to me to join him.

"I've been attending a few conferences and I wanted to share what I believe to be the next paradigm in program management," he said. "There was a particular presentation that focused on leveraging the power of not one, but two standard methodologies, resulting in numerous financial and quality improvements across the board."

I'll admit, that sounded good. During the next forty minutes he described what he had learned the week before.

"I want you to lead the charge on this and implement a hybrid waterfall-Scrum methodology for use on our projects," he concluded.

An especially persuasive ScrumMaster used all his creative charm to convince my executive that our organization would be the perfect laboratory for his new hybrid framework. For nearly six long days the executive had been bombarded to the point of exhaustion with presentations, strategy sessions, and vendor presentations. He somehow determined that the opinion of one individual, absent of any significant show of proof, was enough to warrant the complete overhaul of our product management framework.

## Two Frameworks Walk into a Bar

Any time you smash together frameworks, especially those that aren't similar at all, it is a recipe for failure. Concocting a hybrid methodology sounded as bad as combining DevOps and Scrum into SAFe. Despite my reservations, the decision was made to move forward with this framework, and I was directed to set up the groundwork and begin immediately.

How could a framework that seems to contradict itself be implemented?

Both frameworks have their own strengths. The software development industry leans toward Scrum due to its small footprint, nimble operations, continual feedback, and hypersensitive design. The waterfall approach isn't exactly extinct; some situations still are more suited to the setup of its framework.

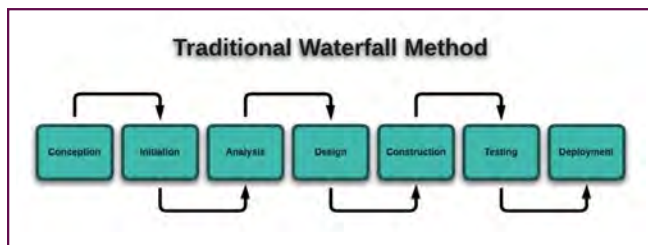


Figure 1: How waterfall works

Figure 1 shows the shell of a waterfall framework.

For the most part, each step follows a sequential path, with each phase usually completed before the start of the next phase.

There is a focus on upfront, long-term planning and little to no interaction with the customer, which adds significant risk of project schedule fluctuations. The ability of the waterfall method to adapt quickly to changes is severely limited by the orientation of how it structures the time between specification and implementation. For predictive projects, waterfall makes a lot of sense. Software development projects, however, aren't usually very predictive.

Scrum, on the other hand, uses an entirely new framework of terminology, including sprints, user stories, ScrumMasters, and product backlogs. The learning curve can be substantial depending on your level of desired mastery. Scrum, as shown in figure 2, takes short-term planning and nimble iterations of project deliverables and consistently gathers feedback from the project sponsors and customers. Its project schedule risk is much lower as it is able to respond quickly to changes due to the short duration of sprints typically employed.

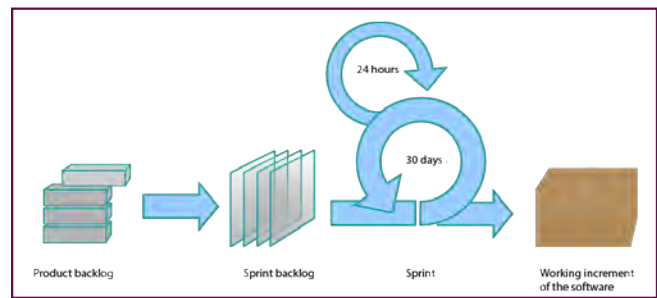


Figure 2: How Scrum works

A product backlog is assembled with the team ordering user stories by priority. The backlog is continually purged into a subset of ordered user stories that are determined to be most important at the time. Over a period of roughly two weeks to a month, the team works on completing the user stories with the goal of producing a workable product. The team meets daily at a scrum standup to remove any barriers and ensure all items are on track for completion by the end of the sprint. Once the sprint is completed, feedback is requested from the stakeholders, new user stories are introduced, the product backlog priorities are re-evaluated, and the team repeats the cycle.

## A Collision Like No Other

The executive expected me to adopt this framework as my own and lead the charge head-on. As the project team was still in its infancy, roughly twelve people piled into the boardroom while I was gathering my courage for our first project meeting. They were mostly a collection of developers, project managers, junior business analysts, and database administrators. Most of them were already familiar with agile, and some with waterfall.

I explained that the program would harness the quality and robustness of the traditional waterfall methodology while ensuring that we adhere to agile design principles and methods. I was more or less following a script about the program's structure handed to me earlier by the executive. As I talked about how this program would harness the quality and robustness of traditional waterfall and the team could adhere to

agile design principles and methods, team members nervously shifted in their seats. I explained the possible synergies that could be achieved by leveraging core competencies across business units while mitigating resource loss inside the framework of a highly controlled waterfall executive committee. Figure 3 was what I was actually secretly envisioning.

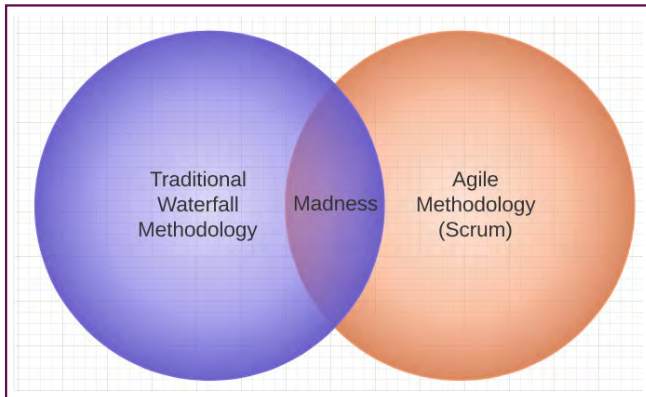


Figure 3: Possible overlap outcome due to integration of waterfall and agile methodologies

To make this work, two primary phases of the waterfall-sprint approach were declared: the macro and micro phases. The macro phase followed a waterfall style and took place over the entire project lifecycle. The micro phase examined the health of the project at incremental product deliverables, similar to what you would expect with Scrum.

The start of the macro phase is called macro conception. The entire project team was assembled to define product scope and identify resource requirements and high-level product backlog items.

These deliverables were added to a project resource sheet, resources were attached to deadlines, and estimates were made. Once this was completed, the team split in two. The macro team acted as a traditional change management control board, while the micro team acted in the traditional Scrum manner.

By driving overall vision and project direction, the macro team acts like a board of directors using the waterfall approach while ensuring the Scrum methodology doesn't move past the position of the current waterfall stage. If the micro team were

about to complete the testing Scrum phase and the macro team had not yet audited those test results, the project would come to an abrupt halt. In essence, waterfall drove Scrum methodology.

The micro team's role is to build the backlog and prioritize work for the upcoming two-week sprints. With the macro team being kept aware of any changes, the micro team operates under the Scrum framework. All project activities require approval from two product champions: the lead of the micro team and the lead of the macro team, as shown in figure 4.

## What Do You Think Happened?

The result of this experimental approach was disastrous. The overhead accompanied by the process was astronomical. Two-week sprints were laughable, as they became distant cousins of Scrum. The absence of one of the major stakeholders in the team at any point in the process meant delays that were significant and devastated deliverables. There were, however, some curious outcomes of mixing waterfall and Scrum.

Product quality from numerous iterations actually suffered. The system had been designed to ensure waterfall-style documentation was administered by the macro team while keeping the nimble design in place for the micro team developers. The unfortunate result was a lot of micro management of sprint product iterations.

Team morale suffered due to the confusing multilevel feedback process that inevitably led to decreased quality. The new waterfall sprints were so inflexible that, although the quality of the feedback from our stakeholders had remained constant, the effort to impact the backlog became bureaucratic, resulting in multiple meetings and change control documentation.

Every time a new piece of feedback that may have altered the backlog was introduced, the macro group appeared to completely renegotiate the priorities of the backlog. What started as a vain and very confused attempt to control the quality of the process outputs ended up decreasing on-time deliverables by 150 percent. Timeboxed iterations morphed into two-week sprints, followed by two-week dead zones that became catch-up grace periods.

To reduce the increased bureaucracy, developers started to combine stories—or not submit them at all. During these dead zones, business analysts took direction from the macro team to direct the documentation that needed to be updated, created, or possibly expunged. This naturally required more meetings,

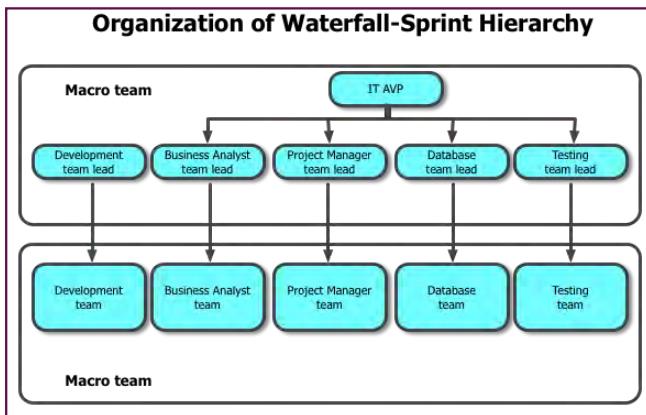


Figure 4: The relationship between the macro team and the micro team

Metric	Scrum	Waterfall sprints
Sprint iterations	2 weeks	4 weeks
Introduction of new item to the product backlog	15 minutes	1-2 weeks
Project schedule risk	Low, adaptive	High, constrained
Time between specification and implementation	Short	Extremely long
Ability to respond to change	High	Nonexistent

Table 1: Comparing the impact of Scrum versus a hybrid waterfall-sprint approach



documentation, coordination, and negotiation. My calendar and the calendars of everyone else on the teams became a block of meetings, back to back.

The final outcome shown in table 1 served as a mechanism to plead with the executives to scrap this program.

After six months of ambiguity, power shifting, and missed deadlines, the program was finally scrapped, with the results summarized in table 2.

Metric	Result
Project budget	28 percent over plan
Critical path	Down 50 percent
Additional documents created	60
Friends made	0

Table 2: Post-mortem results of a failed hybrid approach

The results made for a compelling case to toss the program. The program quickly reverted to the Scrum process, and within two weeks we began to make steady progress toward the project deliverables.

The biggest mistake with applying this hybrid framework was the lack of a single unified approach, a golden rule, to focus the team on. This resulted in two frameworks competing against each other, with neither winning. The product management agile mindset focuses on the ongoing iterative improvement of a shippable product, while project management's waterfall focus assumes a more rigid view of what constitutes the end of the project.

Agile is adaptive; the requirements do not have to be perfectly designed, primarily due to the fact that constant iterations and the continual shuffling of the product backlog allow the focus to be on the continuous improvement of the end product. Product management and project management differ in how resource constraints are addressed. In addition, project management envisions a schedule with definite start and end dates, while product development looks at scheduling as a more fluid concept composed of multiple internal releases, as shown in table 3.

	Product development (Scrum)	Project management (waterfall)
Budget	Quarterly or annually	Allocated once
In-scope activities	Product backlog, increasing (or decreasing) scope gradually from stakeholder feedback	Fixed set of features defined earlier in the process and unlikely to change substantially
Treatment of schedule	Ongoing releases	Allocated over fixed period of time

Table 3: Divergent approaches to managing a hybrid project

Even the definition of an iteration could not provide a

middle ground between methodologies. Table 4 shows a redefinition of waterfall sprints that became a mangled hybrid.

	Waterfall sprints
Budget	Allocated at macro initiation and subdivided by a fixed number of sprints
In-scope activities	A robust set of features created at macro initiation with the intention of minimal change and vigorous change management protocols to restrict flexibility of changing scope without large overhead present
Treatment of schedule	Allocated with a strict end date by incorporating sprints with little flexibility for movement beyond that date

Table 4: Confusion due to applying a waterfall approach to sprint iterations

## A New Manifesto!

The overriding idea of this grand experiment was centered around somebody taking an overall change agent and coordination role between the waterfall and Scrum teams. It resulted in confusion of responsibilities, more meetings, and a proposed Scrumerfall Manifesto that had become a joke among the teams:

- Meetings and processes over individuals and interactions
- Comprehensive unfinished documentation over working software
- Arguing with the customer over customer collaboration
- Following the plan of the plan of the plan over responding to change

## The Long, Well-Traveled Road

Prior to the initiation of this project, the organization had been humming along with an agile framework that had continually produced quality products within the confines of the assigned project resources. The previous successful improvements were implemented using established frameworks routed in research as well as an understanding of the organization's unique culture. Without these components you risk losing employee buy-in and—especially in this case—overestimating the benefits from the associated costs.

This story serves more as a reminder of the dangers of methodology mashing, framework consolidation, and listening to supposed experts instead of basing decisions on known facts. Why fix what is clearly not broken? If it is broken, there are no varying shades of scrum you can blend with other methodologies without needlessly confusing the process. A big benefit of Scrum is how simple the concept of a shippable increment is. If it's not perfect, a revision will come down the line in the backlog, but at least we can ship.

With Scrum, there was significant revision and clarification of the original design to tighten weak concepts or emerging problems. Agile methodology relies on peer review to ensure the work reflects the collective intelligence of more than one or two supposed experts. Before designing your own framework, consider employing a robust vesting process—or even better, *don't design your own framework!* **{end}**

steven.zachary@aleph-one.ca



# *Monetization 2.0*

## *The Evolution of Software Licensing*

*by Omkar Munipalle*



**T**he rapid adoption and proliferation of cloud computing is probably the single most disruptive technological force of this millennium so far. The cloud, along with advancements in mobile technology and Internet of Things devices, promises to radically change forever the way business is conducted. This change spans the entire software industry value chain, including development methodologies, software delivery, new business models, and customer expectations. Needless to say, this has, in turn, transformed the way companies monetize their offerings and go to market with their software solutions.

Software monetization is the adoption of measures a software company takes in order to protect and increase the revenue of its software intellectual property. Historically, companies used hardware devices (dongles) to protect their software from unauthorized use. Although this was effective, it was cumbersome and limiting, especially to users. Software monetization technology and methodologies are now evolving from simply focusing on license protection to both protection and profitability.

The software application market is more crowded than ever. For almost every commercially available software app, the trend is to enable editions on a wide range of platforms: an on-premise version, a cloud-delivered version, and a software as a service version. Software providers are looking at not only every possible avenue to grow revenues from their current client base but also ways to expand into new markets. The next generation of software monetization is not limited to protection and licensing. It's about growth and enabling new business models.

## Welcome to Software Monetization 2.0!

There are six key tenets that are now driving the evolution in software monetization:

**1. Cloud service delivery:** Historically, software was delivered on premise (usually on a server or desktop computer), and the most common licensing methodology was to secure the application with a hardware key or dongle. Then the next evolution was a software-based license key. In addition, software companies could require a license server to be installed in the client environment to secure its software. Software companies are now deploying a hybrid approach to deliver software, running software as a service and delivering the same application on premise (e.g., Microsoft Office 365 and Adobe Creative Cloud).

Next-generation licensing solutions need to be able to handle these multiple delivery methods as well as multiple license models with a single back end utilizing one license enforcement technology. Most software companies struggle to reconcile the different licenses sold to customers and the renewal dates, duration, capacity, and so on. Their software monetization solution should evolve to become the single version of truth about which software each customer has and how much that software is being used.

**2. New business models:** The ability to generate new business models and incremental revenue mechanisms is probably

the most important benefit of adopting a next-generation software monetization technology and process.

In the early 2000s, software companies charged their customers once up front for the software and then added a maintenance contract that was roughly 15 to 20 percent of the up-front license fee. With the arrival of a subscription economy, recurring revenue models are gaining traction and becoming more popular. These models include time-based subscriptions as well as consumption-based or pay-per-use subscriptions, such as number of users, data consumed, bandwidth used, and so on. The next generation of business pricing models will most likely be hybrid models consisting of a combination of subscription and pay-as-you-go models.

This hybrid approach will give rise to à la carte pricing scenarios where companies will have the ability to charge for additional features or more bandwidth as needed. This is going to drive a new monetization channel that is the incremental revenue stream. The true power of packaging can be leveraged here to build once and deliver to many. Feature-based packaging concepts would allow software publishers to enable features in real time through in-application messaging, promotions, and purchases. This flexibility enables the capture of consumer surplus through a low-friction, incremental revenue generation model, a trend that is common across gaming vendors and is seeing wider adoption in the online software services space. The best licensing solution should have the ability to manage, modify, and update entitlements in real time, as needed.

To enable consumption-based pricing models, software publishers need to be able to generate and track true application usage. Some of the analytics that are possible are truly revolutionary. If you correlate application usage with the price a customer is paying, you can quickly derive insights resulting in up-sell opportunities. Key factors include monitoring high consumption at a low price or high price point with low software utilization (churn risk). For example, have the sales team call on a high-churn-risk customer to ensure adoption increases, or begin an email or video training series that educates users on specific features.

Hardware vendors whose primary business model was to price per box are now embracing software monetization concepts to increase revenue. Interestingly, most hardware vendors are realizing that the hardware components of their products are becoming commodities, and it is the software where innovation is taking place, thus giving them the ability to more fully monetize and drive incremental revenue, too. By tracking usage, device manufacturers now have the ability to move from a capital expenditure to operational expenditure model. This opens the door to new markets for expansion, especially where price sensitivity is high and the ability to make high up-front investments is low.

**3. The shift into mobile:** Doing business on a wide variety of mobile devices has become a reality. Many publishers now offer either a stripped-down version of their application—predominately dashboards with similar workflow steps—or browser-based solution that enable access to the application.

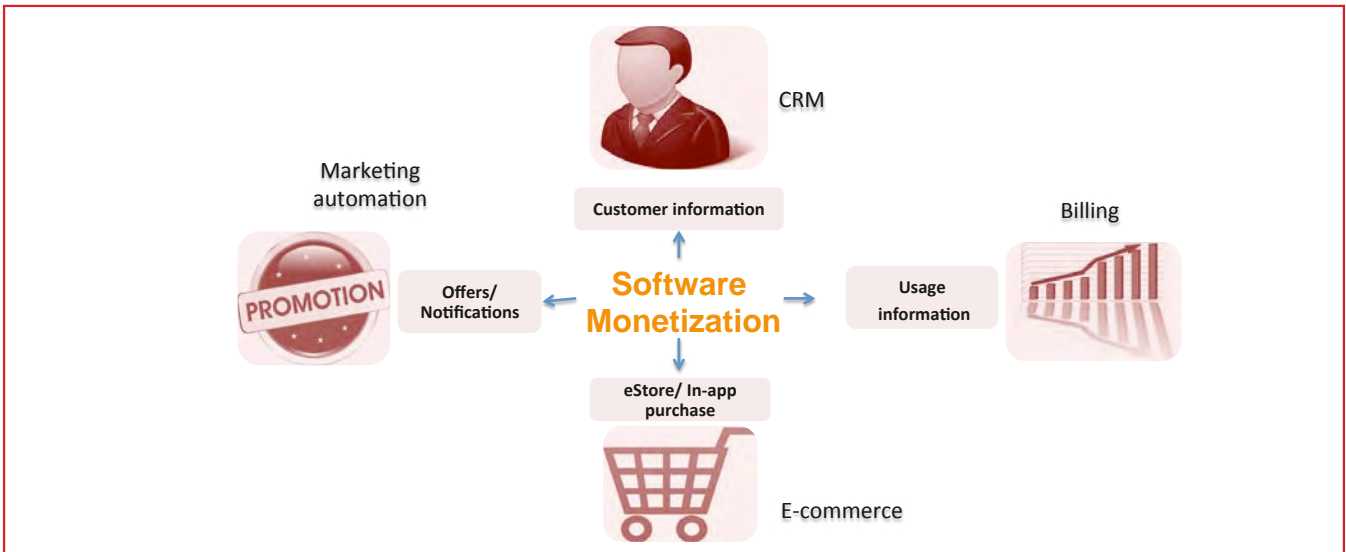


Figure 1: Technology components that need to be integrated into software monetization

There is also a cultural shift taking place. Users are becoming more comfortable with bring your own device cultures and expect continuity in the user experience when they shift between devices to access an application or functionality. The right monetization solution should not only be able to secure the application across multiple devices and platforms (Android or iOS) but also provide a smooth transition between devices. This demonstrates that it is more important to have a user-centric licensing strategy as opposed to one that is device-centric.

**4. Virtualization:** While virtualization has become mainstream, some of the licensing solutions existing today do not perform very well in a virtual machine (VM) environment. Most licensing solutions lock software to physical devices, and in a virtual environment where the machine fingerprint can dynamically change, the solutions fail. Either there is revenue leakage or the application access is denied, detrimentally affecting customer experience. In the scenarios where software vendors want to use consumption or pay-per-use models, most solutions again fail to capture usage at an individual level and then aggregate across the landscape to provide true usage information irrespective of whether a VM is spun down. The ideal software monetization solution should have the ability to function optimally in a virtualized environment and also track consumption to enable pay-per-use pricing models.

**5. An economy of acquisitions:** In the current economic climate, growth by acquisition is becoming a big driver. With the inflow of investment from the market, there is a constant stream of acquisitions and the occasional merger. The speed of integration of the product stacks across companies is one of the keys to success. Having a software monetization and licensing solution that is flexible and scalable contributes significantly to recognizing revenues for the newly combined entity. Often because of disparate licensing and IT systems, it takes a long and sometimes unprofitable time to drive synergies between the companies. Thus, the next generation of software monetization solutions should not only be able to handle scale but also be easy to configure and deploy across product lines.

**6. An automated back office solution:** Time to market is an important parameter in defining the profitability of a product. Given that any product in its current state of functionality has a limited time of revenue generation, the importance of bringing products to market faster is high. An often-ignored secret weapon is the complete automation of the back end with the entitlement management system to reduce the quote-to-cash cycle. Your licensing solution should be able to integrate with your enterprise resource planning, customer relationship management, and billing systems in order to generate a seamless order flow across the relevant IT systems.

The ideal software monetization solution should leverage the services available in the cloud. This reduces the technology footprint and preempts the need to deploy license servers in the end-user landscape. Licensing delivered as a service enables near real-time updates and notifications to the end-user.

Figure 1 shows some of the other technological components that need to integrate with the licensing and usage-tracking software monetization technology, including enterprise resource planning and customer relationship management software, billing, marketing automation, and e-commerce.

Software licensing has evolved significantly in the past couple of years and has reinvented itself from being “nice to have” to “a must-have.” In fact, many software companies now have dedicated teams working on revamping their software monetization strategy. The good news is there are exciting times ahead with the growth of the Internet of Things. From connected cars to connected homes, software monetization is going to further evolve in the coming months to address this clear and present need. An additional consideration is that no software monetization technology should unnecessarily inconvenience the customer. This is an area where embedded licensing, cloud licensing, and software licensing are going to come together in one solution to generate significant value to the ecosystems. Maybe it is time to start putting together thoughts about software monetization 3.0. **(end)**

Omkarnath.Munipalle@safenet-inc.com



### CollabNet Announces the Release of TeamForge 7.2

CollabNet announced the release of TeamForge 7.2 featuring an Open ALM Platform that integrates with individual teams' favorite point tools, has Flexible Process Templates that enable scaling of company-wide tool chains and repeatable processes, and a collaboration architecture that allows mapping of business lines and enterprise technology architectures into categories, groups, and projects.

The new version of TeamForge includes agile planning and tracking features to help make teams more effective, and a deeper integration with Git/Gerrit helps teams securely adopt the most popular open source SCM tool. For organizations looking to scale Agile and DevOps initiatives, TeamForge provides visual velocity diagrams and charts, and specific metric reports for forecasting and tracking enterprise-wide progress.

<http://www.collab.net>

### Xebialabs Releases Updates to XL Deploy, XL Release

Xebialabs announced new features and enhancements to its XL Deploy and XL Release products. XL Deploy and XL Release respectively provide plug-and-play application release automation, and enterprise release pipeline orchestration for DevOps and Continuous Delivery. Both are components of XL Platform, which also comprises XL Test for Agile test management and analysis, and XL Scale for on-demand provisioning and scaling of full environments.

XL Release dashboards have been expanded to include a new, configurable Value Stream Analysis that automatically highlights areas of your pipelines and release processes that are prime candidates for improvement. It also adds a new dashboard that provides a "one stop shop" view of current release and pipeline performance, risk and efficiency.

The XL Deploy plugin API has a new, fully backwards-compatible, rule DSL on top of the existing plugin API, a preview version of the new UI extension framework, and now provides an additional set of advanced reports.

<http://www.xebialabs.com/products/xl-platform>

### DataNumen Releases DataNumen Outlook Repair v. 5.1

DataNumen announced the release of DataNumen Outlook Repair v. 5.1, a business program that lets Windows users scan damaged Microsoft Outlook PST files and recover mail messages, folders, and posts. The application also recovers appointments, meeting requests, contacts, distribution lists, tasks, task requests, notes, and journals.

Improvements in version 5.1 include support for 64-bit Outlook. This allows the software to repair huge PST files and utilize up to 4TB of memory. DataNumen Outlook Repair helps business people recover Outlook files with lost or forgotten passwords. The application also recovers Outlook items that were deleted accidentally.

<http://www.datanumen.com/outlook-repair>

### proDAD Updates Popular ProDrenalin™ Video Optimizer Software

proDAD has announced an update to their ProDrenalin video optimizer. The newly released update of ProDrenalin includes camera profiles for removing unwanted fisheye distortion (warp) from the newest action and sports cameras from GoPro, Contour, Drift, Garmin, HP, ION, Sony, and others. Additionally, ProDrenalin supports the built-in cameras on DJI Phantom Vision 1 and 2 aerial drones.

ProDrenalin is a software utility that cleans up videos shot on action cams, aerial drones, and DSLR cams to make them more watchable. The nature of video shot on these kinds of action & sports cameras often means the video has a lot of visual problems including fisheye distortion from the wide-angle lens these cameras tend to have, lots of shake and vibration, washed-out color, graininess, and camera mounting issues.

ProDrenalin software offers solutions for fixing these videos and exports them in industry standard formats for easy sharing or for importing into video editing software applications for full-blown movie making.

<http://www.prodrenalin.com>

### PDFToolkit 2.0.0- Handy PDF Manipulate Tool from Cisdem

Cisdem has announced the release of PDFToolkit for Mac, a new Mac utility to merge, split, compress and extract PDFs.

Cisdem PDFToolkit provides a set of utility functions for better working with PDF files. It lets users combine multiple PDF files into one, split one big PDF file into smaller ones, compress entire PDF file into smaller size file, extract images out of the PDF, or extract text from the PDF file.

<http://www.cisdem.com/pdf-toolkit-mac.html>

### PowWow365 Releases iPhone App for Meetings On-the-Go

PowWow365 announced the release of an iPhone-led version of PowWow365, its new all-in-one, telco-friendly, feature rich meetings application for scheduling and hosting meetings. PowWow365 combines all the features of Communication and Collaboration (C&C) including VoIP calls, chat, file sharing, along with integrations with third party SaaS applications such as Yammer, Salesforce, Chatter, DropBox, Box, Asana, Podio, and others.

PowWow365 allows users to send invitations directly by accessing the contacts lists, access, control and present pre-loaded documents to attendees, browser sharing and viewing and utilize a virtual whiteboard. Meeting attendees can join a meeting without having to set up a user account.

The app is available for download and installation from the Apple App Store.

<http://www.powwow365.com>

# FAQ

expert answers to  
frequently asked  
questions

by Janet Gregory

janet@agiletester.ca

## The Danger of Testing “Only” Stories

Imagine: Your team has adopted a new tool that contains all your user stories, tests, and defect tracking information for each iteration. The tool makes it easy to map your process with your definition of done to enforce that all defects must be fixed and all tests must pass for each story before the story is considered “done.” The team does all the right things: takes the stories, creates the tests first, codes, and then tests (usually with test automation and exploratory testing). The process appears to work. You and your team must be doing something right!

Why, then, are you still finding defects so late, either right before you release or, worse, when the customer is performing user acceptance testing?

Finding defects late is a common issue when teams fail to remember levels of precision or detail. First and foremost, there is the overall system, or in smaller cases, an application, to which we add new features. Features add capabilities to the product—the business value that matters most to the customer. These features are subdivided into multiple stories, and delivery teams use the stories to implement this business value in an incremental way.

When we consider testing, we should start at the beginning in order to understand the business value of the feature being implemented. As you drill down into the details of the feature and then the stories, consider how the story relates to the feature and how the feature relates to the release. It is vital to understand the relationship and dependencies.

Should you have to interact with other teams to receive or provide information? What are the quality attributes to consider for fit and fitness of your product?

Taking into account the bigger picture and how stories and features fit into the system, the larger the system, the more complicated the interactions are. Starting with business value, let’s define an example of a high-level feature and process:

The high-level feature: `Take customer order.`

The high-level process might look like this:

(Inputs) `Product, customer, quantity` → `[take customer order]` → `place order` (outcome)

Examining the high-level process would be the first place to determine what to test. For example, there are several high-level assumptions that could be verified, including valid customers, valid products, and available stock, and what is meant by each of those phrases. Here are some sample options:

- Customers: could be single buyers (online or in-person), individual companies, wholesale companies
- Products: single products, multiple products
- Orders: regular price, discounts for large orders, wholesale orders, online discount

At the beginning of the release cycle, ask questions to get a common understanding about what is expected. The next step is to determine what features add the most value and should be delivered first. The priority features could be:

1. A single online customer can order multiple products
2. A single company can order multiple products by phone

Mind maps are a good tool to explore features, while flow diagrams can be used to break up each feature into testable stories. As a team discusses each feature, it should create high-level acceptance tests to define the scope of the feature. By taking the first feature, an expected behavior could be something like the following:



## FAQ continued

GIVEN a registered individual shopper is logged into the website  
WHEN she orders more than one available product  
THEN the order is placed  
AND for each product ordered, the availability is decreased by the number of items ordered

It is very important to think of examples of misbehaviors, too:

What happens if a shopper tries to order more items of a single product than are available in the inventory?

When you dig into a complex feature, there will most likely be more than one story. These stories will have interactions with the systems as a whole, but possibly with other features too. Using the idea of steel threads, individual stories might be very specific: one customer, one product, one order. [1] However, the whole feature is a combination of stories, and there is more interaction with other features and the whole system. When teams forget this, they miss the bigger picture. As soon as the feature has been completed, get the customer to try the feature in user acceptance testing. Don't wait until the end of the release cycle—the goal is to receive fast feedback.

If teams have figured out how to test small stories and get them to “story done,” it is time to take a step back and look at the bigger picture. First, consider what it means to be “feature done” and what kinds of tests need to be run to receive feedback as quickly as possible. Step up another level and now think about the overall system. Are there tests that can be run during your iteration to make sure integration issues are found as early as possible?

Remember the big picture—even while testing the small stuff. **{end}**

## NEWSLETTERS FOR EVERY NEED!

Want the latest and greatest content delivered straight to your inbox every week? Have we got a newsletter (or four) for you!

*AgileConnection To Go* covers all things agile.

*CMCrossroads To Go* is a weekly look at featured configuration management content. *StickyMinds To Go* sends you a weekly listing of all the new testing articles added to StickyMinds.com. And *TechWell To Go* features updates on the curated software

development stories and blog posts that appear each weekday at TechWell.com.

Visit [StickyMinds.com](http://StickyMinds.com), [AgileConnection.com](http://AgileConnection.com),

[CMCrossroads.com](http://CMCrossroads.com), or [TechWell.com](http://TechWell.com) to sign

up for our weekly newsletters.



# Leveraging Automated Testing to Improve Product Quality

Test automation is more than scripting test cases. There is a direct correlation between how fully tests are automated and product quality.

by **Rajini Padmanaban** | [rajini.padmanaban@qainfotech.com](mailto:rajini.padmanaban@qainfotech.com)

Independent testing has been gaining a lot of visibility during the past decade. The focus has been on having a team—independent of the one that did the coding—perform validation and verification to ensure product quality. Product teams through the years have understood how to draw the right balance between maintaining independence in quality assurance and collaborating with test teams as and when needed. This is heartening to see. By tightly integrating the interactions between developers and testers, a combined quality focus can greatly help improve product quality while reducing the overall cost of attaining that quality.

## Automating Build Verification Tests

Unit testing has traditionally been performed by developers to validate modules of code to catch issues early in the development lifecycle. Building on developers' testing and troubleshooting skill sets, there are other areas they could take on, given the constraints of time, cost, and quality that most teams operate within.

One area being increasingly looked at is the build verification test process. When a build has been deployed and handed off to the test team, typically the first activity consists of a series of smoke or sanity tests designed to verify that the build is functional and is ready to be further tested. Especially early in a project lifecycle, there are glitches at this stage, forcing the tester to send the build back to the developer to mitigate any issues encountered.

This can turn into an iterative process until the build satisfies the minimum requirements prior to running further tests. To minimize all of this overhead, build verification test automation is a good area for developers to collaborate with testers. Test automation can be effectively used here to verify a build's readiness even before it is handed off to the build team. Due to differences between the developer and tester environments, the test team may have to work with developers to customize the test automation suite to take care of any such differences in configuration and setup. With an increasing adoption

of continuous delivery and integration in agile lifecycles, a lot of time and effort can be saved if developers and testers collaborate in building a sanity test automation suite that verifies the build up front.

## Automating Defect Regression Tests

An important class of validation is defect regression, which is a process that can be fairly simple in some cases, yet very complex in other cases. This approach enables a tester to refer the developer to a set of automated regression test cases that can be run after a defect has been fixed. Defect regression not only takes care of the specific test that the developer would have had to perform, but due to the fact that it often needs to be repeated, it also can save time, effort, and cost, benefiting the efficiency of the entire team.

Consider the situation where a developer hands off a build with fixed defects to the tester only to find that the issues have not been completely fixed. The triage team could be brought in to review the issues again, resulting in the assignment of the defect back to the developer. In agile releases where the team's work is timeboxed, such expensive regressions can significantly deflate the entire team's morale. On the other hand, if the tester provides small suites of automated tests for defects reported, these repeatable tests enable the developer to easily replicate the issue, verify the scope of the fix, and mitigate the issue.

## Other Areas Where Test Automation Makes a Difference

The tester can help the developer leverage automated tests in a number of ways:

- Build a small performance test suite to compare benchmarks needed that enables the same test to be run again and again.
- Create a test code pack that enables a form of code scanning for analysis.

“Automating tests offers significant benefits to developers because they can leverage the tester’s repository of automated tests.”



- Leverage a simple suite of end-to-end test cases the developer can use to evaluate product quality.
- Identify a set of tests that validate compatibility, especially if the product is cross-platform.

Automating tests offers significant benefits to developers as well because they can leverage the tester's repository of automated tests to catch issues early, saving time while, more importantly, improving product quality. The benefits, such as bringing repeatability to an effort and greater accuracy that helps bring down human error, are benefits that automation would provide not just to a tester. These are benefits that a developer can benefit from as well, for whatever minimal testing they take on.

### How Practical Is Sharing Automated Tests?

While there are significant benefits to sharing automated tests, the commitment to invest in such a collaborated test automation effort is easier said than done. It requires a lot of maturity in the product team to understand that the tester is not delegating his role to the developer, but rather that this effort is only helping improve the overall health of the product

and optimize processes. The tester has to understand that it is not about just a one-time handoff of the automated tests to the developer. He will need to collaborate with the developer on an ongoing basis to see what maintenance the test suite might need, any customizations that need to be performed, and any help the developer needs to use the tests with minimal effort on his part.

In an ideal scenario, the tester will be doing all the heavy lifting by building a solid test automation framework and then building tests on top of it. The goal is that these tests should be modular, require very minimal maintenance effort, and encourage the developers to make minor tweaks by themselves. The tester has to clearly know where to draw the limits so that the developer takes on just the initial vetting with automated tests, while the actual quality assurance responsibility is still primarily with the test team.

When this level of quality maturity is incorporated into the very fabric of the project team, it not only improves product quality. It also establishes healthy relationships among the entire team, demonstrates that quality is everyone's responsibility, brings a new awareness to what quality is all about, and raises the importance of the impact of quality to the end-user. **{end}**

## index to advertisers

Agile Development & Better Software Conference East	<a href="http://adc-bsc-east.techwell.com">http://adc-bsc-east.techwell.com</a>	9
ASTQB	<a href="http://www.astqb.org">http://www.astqb.org</a>	17
Call for Speakers	<a href="http://vlt.me/speaker">http://vlt.me/speaker</a>	1
Capgemini	<a href="http://www.capgemini.com/testing">http://www.capgemini.com/testing</a>	Back Cover
Mobile Dev + Test Conference	<a href="http://mobiledevtest.techwell.com">http://mobiledevtest.techwell.com</a>	Inside Front Cover
Ranorex	<a href="http://www.ranorex.com/whyBSM">http://www.ranorex.com/whyBSM</a>	2
SQE Training	<a href="http://www.sqetraining.com">http://www.sqetraining.com</a>	8

**Display Advertising**  
[advertisingsales@sqe.com](mailto:advertisingsales@sqe.com)

**All Other Inquiries**  
[info@bettersoftware.com](mailto:info@bettersoftware.com)

*Better Software* (ISSN: 1553-1929) is published six times per year: January/February, March/April, May/June, July/August, September/October, and November/December. Back issues may be purchased for \$15 per issue plus shipping (subject to availability). Entire contents © 2014 by Software Quality Engineering (340 Corporate Way, Suite 300, 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 (Software Quality Engineering). 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.



# Taking mobility testing to the next level



Led by Capgemini's mobile solutions and testing experts, our global Mobile Testing Center of Excellence is equipped with the latest testing tools and infrastructure from industry leading partners plus proprietary tools and accelerators developed by Capgemini. Our CoE acts as a hub for mobile testing services and brings together people, tools and processes to support mobile testing engagements.

## What We Do

The CoE delivers performance, functional, compatibility, usability and security testing for mobile environments. With global penetration of mobile devices at 93% and growing\*, firms must ensure the mobile enterprise strategy takes testing into account right from the start. Capgemini uses a wide variety of mobile testing tools and addresses the challenge of platform fragmentation and usability testing through a core set of physical devices covering all major device families and operating systems. This infrastructure is combined with a private cloud solution so mobile testing can be run from any Capgemini location across the globe.

With change a constant in the fast-paced mobile industry, we keep a strong focus on research, development and innovation. As new tools and platforms emerge, we create studies, methodologies, and proofs of concept for new mobile testing tools to spur innovation and meet client challenges. Capgemini supports mobile development and testing for some of the largest financial firms in the world. We have strategic alliances with technology leaders including IBM, Microsoft, Apple, Google and SAP.

\*Source: Social, Digital & Mobile Around the World, January 2014, Wearesocial.sg

## Need More Information?

Learn more about our five mobile testing services, Mobile Testing Center of Excellence and overall mobile and testing solutions:

- Mobile Testing Services
- Testing Services
- Mobile Solutions



## About Capgemini

With more than 130,000 people in 44 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2013 global revenues of EUR 10.1 billion. Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

*Rightshore® is a trademark belonging to Capgemini.*

People matter, results count.