SHOWCASING THOUGHT LEADERSHIP AND ADVANCES IN SOFTWARE TESTING

LogiGear Magazine

TESTING SMAC DOWN

LEVERAGE MOBILE TESTING WITH TestArchitect™

WHAT DOES SMAC MEAN FOR SOFTWARE TESTING?

- 8 LESSONS LEARNED IN MOBILE APP TESTING
- SMAC 101
- SMAC AND THE HISTORY OF IT
- CONTINUOUS TESTING, PART 2

SEPTEMBER 2016
VOL X
ISSUE 3

Editor-in-Chief
Michael Hackett

Deputy Editor
Christine Paras

Graphic Designer
Tran Dang

Worldwide Offices

United States Headquarters
4100 E 3rd Ave, Ste 150
Foster City, CA 94404
Tel +1 650 572 1400
Fax +1 650 572 2822

Viet Nam Headquarters
1A Phan Xich Long, Ward 2
Phu Nhuan District
Ho Chi Minh City
Tel +84 8 3995 4072
Fax +84 8 3995 4076

Viet Nam, Da Nang
346 Street 2/9
Hai Chau District
Da Nang
Tel +84 511 3655 33

www.LogiGear.com
www.LogiGear.vn
www.LogiGearmagazine.com

Submission guidelines are located at:
http://www.logigear.com/magazine/calendar/2016-editorial-calendar-and-submission-guidelines/

LETTER FROM EDITOR

As we settle into autumn, we're taking the time to start some new traditions. This is LogiGear magazine's first issue on SMAC. SMAC—social, mobile, analytics and cloud. We will be doing more issues in the next few years on these topics since so much of the product world is moving to this development stack.

There are many ways that we can talk about SMAC. SMAC is seen as the natural evolution of mobile testing. Since mobile has evolved into a platform, products are being introduced rapidly. From enterprise apps, games, banking, brokerage and financial services with many using *Internet of Things* devices: mobile no longer only means mobile phone; it now also means mobile control device.

We also know now that the number of APIs available to be incorporated into products continues to grow exponentially. Most organizations, including test teams, saw this on the horizon and were hopefully preparing for this inclusion in their product lines.

The biggest challenge I see for test teams is how quickly the majority of our work has moved to the cloud. Many tools are not ready for cloud use, and many teams don't yet understand the differences with testing in the cloud, or testing cloud services, or cloud stored data. It may take some teams awhile to see the ramifications of moving the testing environments and production environments outside of the organization's control into the cloud. The cloud will impact in a very positive way, how easy it is to get access to great testing environments. The biggest selling points for cloud adoption are the full, quick and easy access to any kind of environment you want on demand—staging environments, production environments—all of these are now on demand.

The great unknown in the SMAC stack is analytics—-at a minimum for test teams the use of analytics will impact how testers design tests. Introducing various technologies like A/B testing and dynamic product design will make test case design more fluid and *just in time (JIT).*

This is truly an amazing time to be working in product development, from mobile controllers, to the Internet of Things, to the vast array of easily accessible web service/APIs; all this product development rapidly delivered using DevOps development paradigms makes getting new, innovative, and exciting products into customers' hands quickly—with the benefit of connected hardware *things*—an exciting time to be working in software development. With testers in most organizations being the last line of defense for quality before your product gets in your customers' hands—using analytics for product in test design the importance and value of testers in development organizations has never been so great.

Learn a lot about these technologies: they will be here for a while. Have fun!

In this issue, we feature the second part of our Continuous Testing series. Alister Scott has also returned as our Blogger of the Month to tell us about eight lessons he's learned in Mobile App testing, and Daniel Knott warns us about the ten mobile app testing mistakes to avoid. This issue of TA corner features how to use TestArchitect in your mobile testing strategy, and Leader's Pulse discusses how to get your test teams the skills and strategies they need to test in this new wave of SMAC. We also have two infographics for you, one that is essentially a SMAC 101 infographic (what to test, and the how) and a history of IT and how it relates to SMAC.

IN THIS ISSUE

LOGIGEAR ANNOUNCES NEW SOA TESTING SERVICE PACKAGE

For over 20 years, LogiGear has helped organizations deliver SOA testing efforts including SOAP and REST based systems to ensure the robustness, reliability and resilience of the web service. LogiGear helps organizations achieve and keep up QoS they need to stay ahead of competition by implementing a comprehensive SOA testing strategy that covers functional, load & performance, interoperability, and security testing from a business needs perspective.

Read more at: http://www.logigear.com/services/soa-testing.html

IN FIRST FOR INDUSTRY, AUDI VEHICLES WILL COMMUNICATE WITH TRAFFIC SIGNALS

In the efforts to alleviate crashes and reduce congestion in the United States, Audi is introducing technology that will allow vehicles to communicate with traffic signals. The technology will allow traffic signals to exchange safety and operational data with vehicles through the cloud. Audi plans on releasing this ability in about five years, in the hopes to reduce stress and to increase safety.

Read more at: http://www.nbcnews.com/tech/tech-news/first-industry-audi-vehicles-will-communicate-traffic-signals-n631676

XBOX GAME PREVIEW WILL SOON LET WINDOWS 10 USERS TRY GAME EARLY

Microsoft's Windows 10 will soon implement Xbox Game Preview staring Rockfish's Everspace. Xbox Game Preview allows players to test games while they are still in the process of being designed. Although these games are in earlier stages and are more likely to have problems, gamers will get the opportunity to give feedback that could change the progress and development of the game. Microsoft's desktop OS only includes Rockfish's Everspace, but will soon include many more games for gamers to shape.

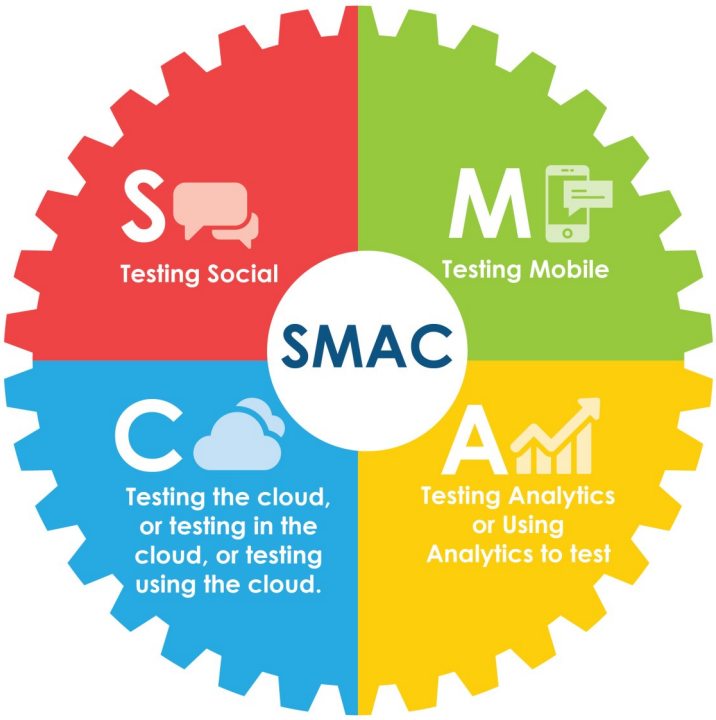
Read more at: https://techcrunch.com/2016/08/16/xbox-game-preview-will-soon-let-windows-10-users-try-games-early/

WHAT DOES SMAC MEAN FOR SOFTWARE TESTING?

BY MICHAEL HACKETT

Much has been written over the past few years about SMAC—Social, Mobile, Analytics and the Cloud—as the new platform with exponential growth. In this article we will answer the question: what is this and what does it mean to software testing?

What is SMAC?



S—Testing Social

M—Testing Mobile

A—Testing Analytics, or Using Analytics to test

C—Testing the cloud, or testing in the cloud, or testing using the cloud

Not only as individual parts of the system but also testing the integrated product or service and adapting that testing based on the output of analytics based on real usage as well as the prediction of use.

To me, SMAC is the natural evolution of mobile as a platform. It is the convergence of these technologies, also called the SMAC stack that is leveraging of each other to build and develop systems and products with great functionality, and deploy them very quickly into the marketplace.

The most common phrase people are using to describe the outcome of SMAC is "greater than the sum of its parts."

SMAC is also coupled with the *Internet of Things*. Since connected, smart devices are flooding into the market, often controlled by mobile devices and data from these devices is often part of the data collected and stored in the cloud for analytics.

This technology use is exploding and getting bigger everyday. If you don't realize how big and mobile the world has become, this article is not for you. Mobile and web services use is growing exponentially every day, making it pointless to toss out numbers and percentages of just how much these technologies are being used, because these numbers get outdated as soon as they are published.

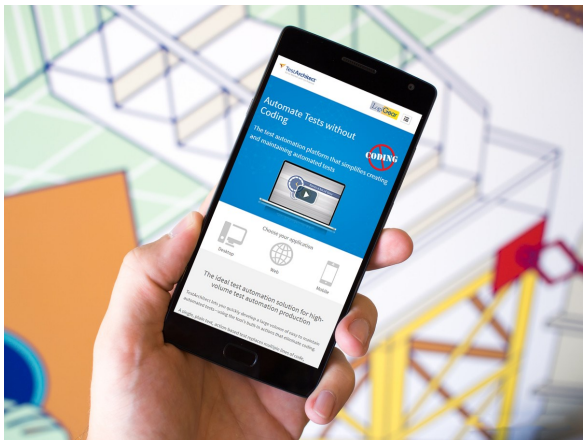
Evolving as a Platform

Not too long ago, the desktop expanded to a web server, application server, database, varieties of web browsers—that was a big jump for teams whose testing had been restricted to desktops. New environments, new services, new technologies to test. From concerns about iOS versus Android and all the varieties of Android, Mobile is a platform.

It's Social. Mobile uses Social Media APIs or web services or Web APIs, or even just using any web service in larger and larger importance every day.

A common example of the IoT has been the smart

home, managing your energy usage remotely from your mobile device with the energy company collecting user and usage data, coupling that into *big data* with weather/temperature data. This data captured from a variety of sources using APIs and web services and smart devices and put it in a data warehouse in the cloud. Obviously, the data has to be kept safe, it has to be ready when systems or people need it and the cloud databases need to respond well and quickly.



The product will do some analytics to make sense of the data. And, as an example broadcast a notification of a "brown out" (reduced energy available or shut down to certain areas during peak usage usually during the hottest time of the year), and is sent out over email and social media, perhaps both Facebook and Twitter.

How many places do we need to inject testing in this example?

We could describe examples in consumer and corporate or enterprise app; banking, finance, real estate, retail, sales and marketing. Banks are now accepting payments through Facebook and Twitter. LinkedIn is being integrated extensively into secure enterprise apps. For people who think SMAC is only consumer apps—you are wrong—the application of this stack is far reaching. Mobile access to corporate internal assets—such as CRM, is now taken for granted.

But for us, in this article, let's focus on the testing aspects.

Terms and tests

Social is most often referred to as the huge social media players but, also includes product integration with the massive number of web service APIs such as integration with Amazon, Pandora, eBay and thousands more. The most common use of social media in SMAC might be the most straightforward to test: real-time engagement and feedback. Correct integration of the various social apps enables these.

Mobile is the device, all the platforms and hardware and applications that goes along with it—cameras, geolocation and location services, for example, features unique to Wi-Fi versus 3 or 4 G. Part of your mobile strategy must include: cross-platform, responsive, location aware, memory, CPU and battery usage to name a few.

Analytics is the most complex, and probably newest to most testers. This is testing the gathering, storage, manipulation, analysis, and use of data for a variety of purposes such as learning and optimizing the user experience or predicting behavior.

There are many, many places that need testing in this aspect of data warehousing—from the algorithms and system availability to performance.

Cloud, simply, is testing the cloud infrastructure the product uses. It's like testing a database server as a remote location for all the things you already know to test here, such as; data integrity, performance, security, fail-over. Databases—with and without big data attributes need their own sets of tests.

Obviously, functionality has to be tested, bugs need to be found, and User Stories need to be validated at each place.

The Central theme of SMAC is that the Integration's the thing.

The Stack presents its own issues.

According to Admonsters when people refer to "the Stack" they mean that the technology stack is a set of components or layers in software offering that

provides broad functionality. Microsoft Office is an easy example. The combination of Word, Excel, PowerPoint and Exchange improves personal productivity for billions of people.

What this means to testing holistically, is that you may have a mobile test strategy; you may have a web service test strategy, etc. But putting these four technologies together into a stack makes them significantly more useful to your company's business, and also significantly more difficult to test—even if only from a scale, skill, and tool perspective.

The SMAC test strategy focuses on security, performance, integration and usability. All these technologies get integrated into one platform while needing seamless usability.

Testing

There will be various interfaces to test:

First, the customer interface: the functionality, usability, performance. This is pretty well understood.

The IoT device interfaces. Testing embedded systems is also well understood but new to many testers.

There may also be secondary customer, internal customer, alternative users, partners who may be providing or consuming data using different interfaces—either different user interfaces or APIs into their own systems.

There will often be unique interfaces for internal employees into the social media integration, as well as interfaces into various places in the data warehouse—raw data, processed data, output data, etc.

Focus on Analytics

The most complex and interesting testing for traditional testers may be the testing associated with the analytics.

There are many aspects of testing here. The data itself needs to be tested and there can be large varieties some examples include: user behavior data, media data, as well as dozens more varieties. It can also come in large quantities. Sensors can be streaming massive amounts of data.

One of the goals with analytics today is for designers, developers (programmers and testers) and the system itself to learn about the users and usage from the data. The more the system is used, the more data it collects, the better the system becomes. This is called machine learning. Many of these SMAC products are built to gather information with the sole purpose of predicting how users will use it and what they will do next. This leads us to a unique position in software development. This business intelligence (BI) is used by us to design tests. We will design tests based on real usage from this data. We will *do A/B testing* where we may have the system designed so we can vary a task or function and measure what users do or prefer as well as build and test varieties of a task or function and swap them in and out based on data collected. This BI exists to impact our test case coverage, automation, and risk decisions.

Things you do with data: collect it, perhaps clean it

up, store it, search it, mine it, analyze it, send it somewhere for you to use, give or sell it to someone else. This needs testing at every step.

Testing the cloud

Cloud testing, in many cases, is similar to traditional server testing. See the *evolving as a platform* section for the type of tests it includes.

Automation implications

There are a few special impacts to test automation. You may need to expand your suite of tools to include tools that test the constituent parts as well as end-to-end workflows and paths. Not many automation tools can do mobile, web service, desktop UI (for data warehouse tests), SQL tests and even run in the cloud. Finding the right test automation tool or tools

is important. To learn more about the criteria you should use, you can refer to this article on test tool selection.

SMAC Development and DevOps

Combined with speed of integrating web service apps and functionality, using cloud infrastructure and the speed of ease and speed of deployment—for most teams, SMAC systems can be built and deployed quite rapidly. This puts an added emphasis on greater team collaboration and early, *shift left* testing as well as effective test automation. Regardless of doing the DevOps practice of Continuous Testing or not—be sure to leverage technology and take advantage of virtualization for test environments and data.

Also, built into SMAC is the data collection and analytics commonly used in the Continuous Monitoring practices in DevOps.

Skills

As you have seen in this article, there is a wide assortment of skills—many new skills that a test engineer working on the SMAC stack will need proficiency in. For many teams this will require a big upgrade in skills and technology. See our Leader's Pulse article for getting your teams these skills.

In addition to the technical skills and understanding the platform for better test design, there are some testing skills that need to be especially strong: defect analysis and isolation, creative test case development, test case development based on analytic data and A/B Testing test case development.

Summary

- SMAC is set of technologies that used together, as a stack, are becoming more and more common for sophisticated and rapid product development. We need a new test strategy to satisfy this demand.
- Social—SOA testing (service oriented architecture with many APIs/web services).
- Mobile—we have been doing mobile testing a few years. The platforms have matured, the tools have matured, and the practices are more common.
- Analytics—this may be new to many testers. The impact of so much data on our testing is a new frontier in our work.
- Cloud—as a test platform or a platform to test may be new to many testers as well.

Michael Hackett

Michael is a co-founder of LogiGear Corporation, and has over two decades of experience in software engineering in banking, securities, healthcare and consumer electronics. Michael is a Certified Scrum Master and has co-authored two books on software testing. *Testing Applications on the Web: Test Planning for Mobile and Internet-Based Systems* (Wiley, 2nd ed. 2003), available in English, Chinese and Japanese, and *Global Software Test Automation* (HappyAbout Publishing, 2006).

He is a founding member of the Board of Advisors at the University of California Berkeley Extension and has taught for the Certificate in Software Quality Engineering and Management at the University of California Santa Cruz Extension. As a member of IEEE, his training courses have brought Silicon Valley testing expertise to over 16 countries. Michael holds a Bachelor of Science in Engineering from Carnegie Mellon University.

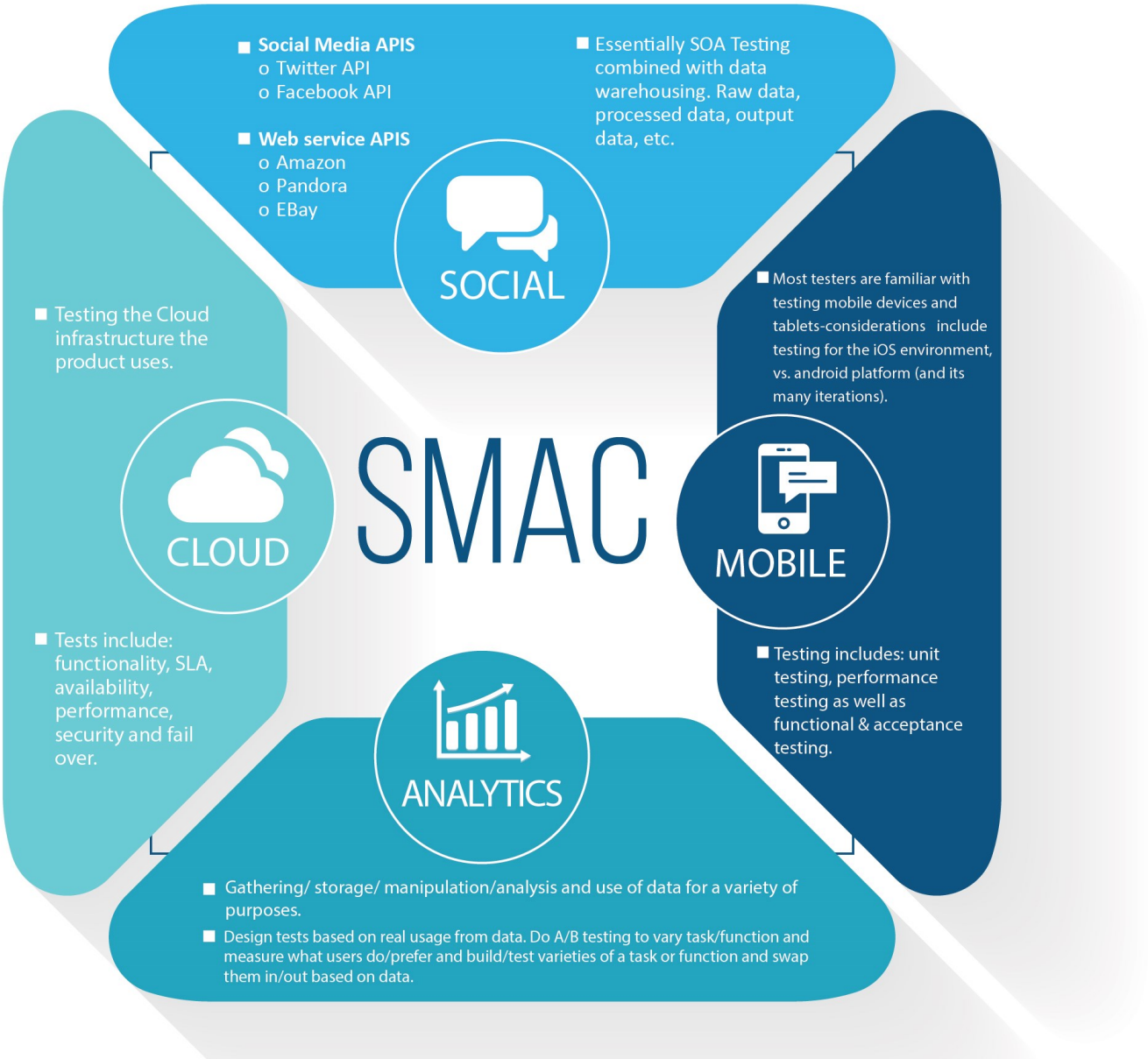
SMAC 101

BY CHRISTINE PARAS

SMAC (Social, Mobile, Analytics, and Cloud) is the natural evolution of Mobile as a platform. It's the convergence of these technologies also called the SMAC stack, that when leveraged, can rapidly become products with great functionality.

SMAC is a set of technologies that when used together, is becoming increasingly crucial to a firm's bottom line. You need to have a test strategy for SMAC.

Most already know about SMAC and its benefits, but having a holistic testing strategy, as well as the expertise for each area is crucial:



INTEGRATED, "INTEGRATION TESTING"

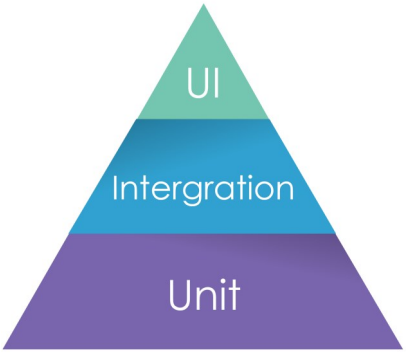
BY JAKE WALLINGFORD

As a software development company, what is your goal? What is the one thing you feel you need to do to ensure you have a job at the beginning of each wonderful work week? The answer is actually quite simple; You need to deliver a quality product. Like how I used the word simple? Although the answer I gave was simple, the process to make that answer possible can sometimes be a frightening, unclear, and glooming reality. But then one day you wake up and ask the question, what can I do to make this better? Well, for those who have woke up and asked themselves that question, this should help clarify at least one step in the process to a smoother and more collaborative product delivery.

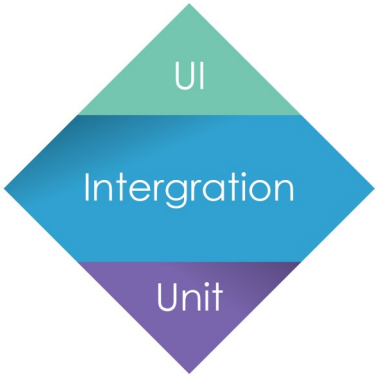
At the core of every great product, is a great team that has the ability to ensure a product delivery that puts a smile on everyone's face. We all want to push a product to production and make sure it happens with the least amount of flaws as possible. We also want to do that as fast and efficient as possible. That is where continuous integration becomes a key aspect in your software development mindset. When you look at the grand scheme of things, what scares you about continuous integration? It usually involves a lot of testing, and if you don't have the proper test automation in place, how will you receive that feedback in a timely manner?

Well, continuous integration requires continuous feedback. What better way to get continuous feedback then a list of test results that solidify stability of an application? With DevOps at the forefront of every innovative software application these days, the need for solid test automation is critical. It's so critical that I would go as far to say that you cannot call yourselves a true DevOps shop without some form of test automation built around the application. You have unit testing, integration testing, UI regression testing, and the list goes on and on. So you ask yourself, what kind of test automation is out there? How much do I really need to automate, and where is my value? Well, as many within our company know, I like to break it down into a simplistic format.

Traditionally, you had your three levels of test automation and they were tiered in a fashion that represented a triangle. A whole bunch of unit tests, a good chunk of integration tests, and a few UI smoke tests. Kind of like such...



Many software development shops use this test automation triangle and it serves them quite well. Tests are still automated and the end result is pretty successful. Why argue with something that works, right? Well, I began to ponder on the efficiency of this model after having numerous discussions with some of my very knowledgeable co-workers. Why not tweak that triangle a little bit, and in the spirit of DevOps, get the whole team involved with the process? Wouldn't it be neat to live in a world where a developer and QA actually worked together to build an automated test? An "integrated", integration test. Sounds pretty cool doesn't it? In a nutshell, that is exactly what we did at my company. Now, before I get into explaining the process, let me do a quick explanation of what our test automation triangle looks like now, and why it looks like it does...



Funny looking triangle right? As our team began to develop and improve our testing processes, we found that the traditional triangle reference quickly shifted to a diamond reference. We began to write fewer, but higher quality unit tests around our code. The integration testing became the bulk of our testing because we were able to test our business logic, separate from the actual UI, in a much timelier fashion. And of course, we still had our UI smoke tests that really confirmed that our site was looking the way it should and functioning appropriately. In the past, a developer would write a unit test or integration test, leaving a large portion of the testing effort on them. With a large portion of the testing effort falling on a developer, that leaves less time to actually develop.

The trickle-down effect starts to show its wear and tear on the unit tests. They become stale, unmaintainable, and irrelevant after a few iterations of work simply because the developer just doesn't have the time to do it all. Not to mention, the QA were completely hands off from our code and left in the dark with many questions when an ugly bug would arise. In reality, our QA also had a lot of good scenario -based input that was valuable to a developer, but often times that got lost in translation. So what did we do? We brainstormed, we kicked it around, and we finally found an awesome solution for getting the team integrated from a testing perspective.

That is where the wonderful tool of TestArchitect came into play. We were able to utilize TestArchitect as a source for data storage when writing automated integration tests with the help of a developer. The way it works is pretty slick. First off, you have a QA who creates a test module within the TestArchitect tool, this is where the test data or test scenarios lived. Next, a developer would code up a test harness that connects the communication from the data in the test module to the data being returned from a service or API. The cool thing was, when you execute the test, you are left with a combined effort to make that test successful. The developer didn't do it all, and the QA provided valuable input on the data scenarios that needed to be tested. We "NEEDED" both the developer, and the QA, to work hand in hand which created a natural collaboration between the two of them.

With the help of TFS, you can then associate all of those tests to run daily, hourly, at check-in, or whenever you feel they are needed to verify that the software being built is being thoroughly tested. I like to explain it in a simple process flow like this; A QA creates a test module that houses scenarios of data within it. The developer creates the harness that links the test data to the service or API that is being tested. And in result, the test executes and shows the quality of your application under test. It turned out to be a really collaborative effort and something that benefited our team tremendously.

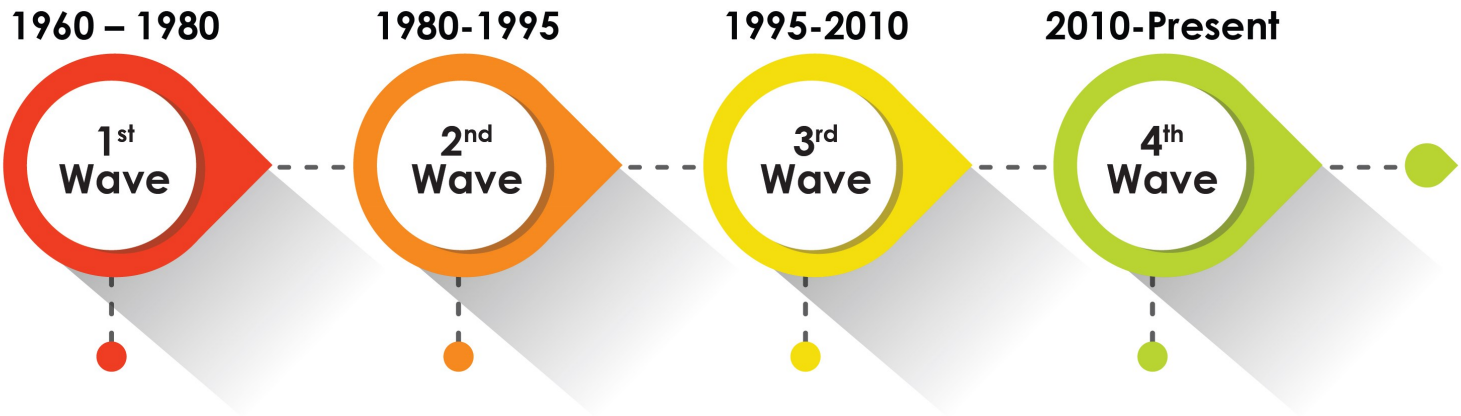
So when I titled this article, "Integrated, Integration Testing", it was intentional. It truly was an effort that integrated our team. With that integrated effort came challenges and experiences that provided so much value to the team in the long run. For the first time, our QA were able to understand the logic within the services of our site, and the developer was able to understand testing from a QA perspective based on the constant exposure to the test scenarios. The more we sat back and watched all of this collaboration happen before our eyes, the more we realized that we have something that really benefits the team. It was really neat to see a team work together to deliver a product that everyone truly cares about. At the end of the day, we all want to build software that pleases our customers. Doing it as an integrated team, just made it that much more enjoyable.

Jake Wallingford
Senior Technical Lead

Senior Technical Lead, specializing in Quality Assurance. Experience, and background in Agile software development, Continuous Integration, Continuous Testing practices, and Test Automation. Passion for testing software and providing the best possible online shopping experience for our customers. LinkedIn: https://www.linkedin.com/in/jake-wallingford-a43842b7

SMAC AND THE HISTORY OF IT

BY CHRISTINE PARAS



Mainframe & Mini Computing

Before the internet, software testing was extremely limited in its capabilities. The majority of testing done was functional, and was done through the mainframe/ server.

Applications were only in the thousands, and users were in the low millions, and there were less than 10 million computers in existence.

Mini Computing PC & Client Server

The 80s saw the dawn of mini computing as desktop computers became the standard, and were eventually replaced with laptops, causing computer use to mushroom to a whopping 10 million.

Testing evolved to now include testing the desktop, client and the server. Performance testing started to become a more common task for teams.

Internet (Web)

With the adoption of the World Wide Web, computer use grew in size, in services offered, in functionality delivered, in supported hardware—in every way.

This grew testing from a nag and a delay in development to the key development step for customer experience and customer retention and satisfaction.

SMAC

The current wave of software and hardware testing has grown to include smartphones and emerging IoT devices, as communication and access to information grows.

Testing will have to grow to include several areas of things that were never previously tested together as one system before, social, data analytics, and this will have to also be done in cloud computing, as companies move to develop, serve and store their applications and data in the cloud.

This infographic was based on Computerworld's article "SMAC and the Evolution of IT"

CONTINUOUS TESTING, PART 2

Strategy and Automation Goals for Test Teams

BY MICHAEL HACKETT

Now that Dev Teams have had a little time to settle into Agile, the new wave of process optimization has arrived. DevOps. DevOps has been described as Agile on Steroids. DevOps has also been described as Agile for Operations/IT. I like both of those descriptions as well as some others. Many organizations want Development, Test and Operations teams to move to DevOps now.

DevOps is a big topic. But DevOps is not the focus of this article. We will not be talking about, for example, containers, or Docker, or Puppet or *Infrastructure as code*. In this article, we are going to focus on Continuous Testing. I will focus on what Test teams are responsible for—the practices and concerns for testers to have their work become an asset to the team and not a drag, in this exciting new world of DevOps!

This is part 2 of a 3 part series on Continuous Testing. ***Part 1*** was on the rationale, business goals, culture change and overview with some specific testing tasks.

Part 2 is a deeper dive into the specific ideas, practices and tasks of test teams that need to be updated or changed to be successful in this next phase of product development. This focuses on test strategy, automation goals. Part 3 will discuss more automation in DevOps/Continuous Testing issues as well as environment and data problems to solve, virtualization, with more details on testing goals.

We gave a quick overview of the Continuous Testing Process in ***Part 1***.

1. Start with an automated smoke test. Move these into CI build process tool.
2. Build bigger regression suites. Move these into CI build process tool.
3. Grow in *levels of awesomeness* of CI; run smoke and/or regression on multiple VMs or in the Cloud.
4. Easy and effective reporting back to the organization.
5. Use containers and/or virtualization for data and full production-like environments.
6. Distribute automated tests into different suites with varying goals on different environments. Use VMs for various environments to grow automation, coverage, speed, monitoring and feedback to the team.

As we get into this discussion, there is an additional idea to remember.

DevOps is about delivering value to the customer when the **business wants and needs it,** not when Dev or Test or Ops *gets* to it.

Now, let's go into more detail.

These topics are very closely related:

- Continuous Testing Strategy
- Examine and redefine regression
- Rethink your automation

Strategy

Continuous Testing is not merely testing continuously. It's more intelligent than that. A very common mistake is to run and rerun the same *full regression suite* on different environments and think that is enough.

Instead, you need to think of what and why we test on different environments.

What Environment?	Dev	Testing	Staging or Pre-production	Live or Production
Who	**Developers**	**Test team**	**Test & Ops**	**Test & Ops**
Purpose	Introduce new code, test in isolation, test integrated code, small data, not fully functional	Test through UI, full system, integration on limited environment, bug finding, experiment, discovery of how system works	Production-like, Fully enabled, every piece of the system in pace, policies, bigger or full data, no "traffic"	Monitor live system
Tasks	Unit testing, component, integration, API	UI, End2End, workflow, regression, bug verification, system testing small, limited data, API Acceptance/ Validation	Validate system on environment as close to production as possible.	Live, immediate feedback, monitoring
Automation	Unit API Smoke	Smoke Acceptance API UI Functional Regression	End2End user focused tasks Happy Path	Monitoring verifying live system operation

For example, what tests need to get run on the build environment? Why? What tests get run in the test environment? Why? What automated tests get run against, for example, VMs with various configurations, languages, browsers, mobile devices, etc.? What automated tests get run on the staging server? What is their purpose? What gets run on the staging server? What gets run in production?

In Agile/Scrum, Dev/Test teams *progressed* through these environments. In DevOps/Continuous Testing, you might be running your automation on many or all these environments concurrently.

Change your understanding of Regression

Regression as a term is probably the most misunderstood term in all software development.

A Regression test is any test you run again. Some people in development think the regression suite or full regression is running every test run against a previous build or product. That is rarely the case. The regression suite is more likely rerunning every automated test. Well, not every test is automated, so only the automated tests are run again. Or regression could be *happy path* tests, which probably means only the main user tests POs must work for a customer to be happy. Not boundary or edge or rare or bug-finding cases—only the perfect, easy, no problem tests. How much confidence do you get from these?

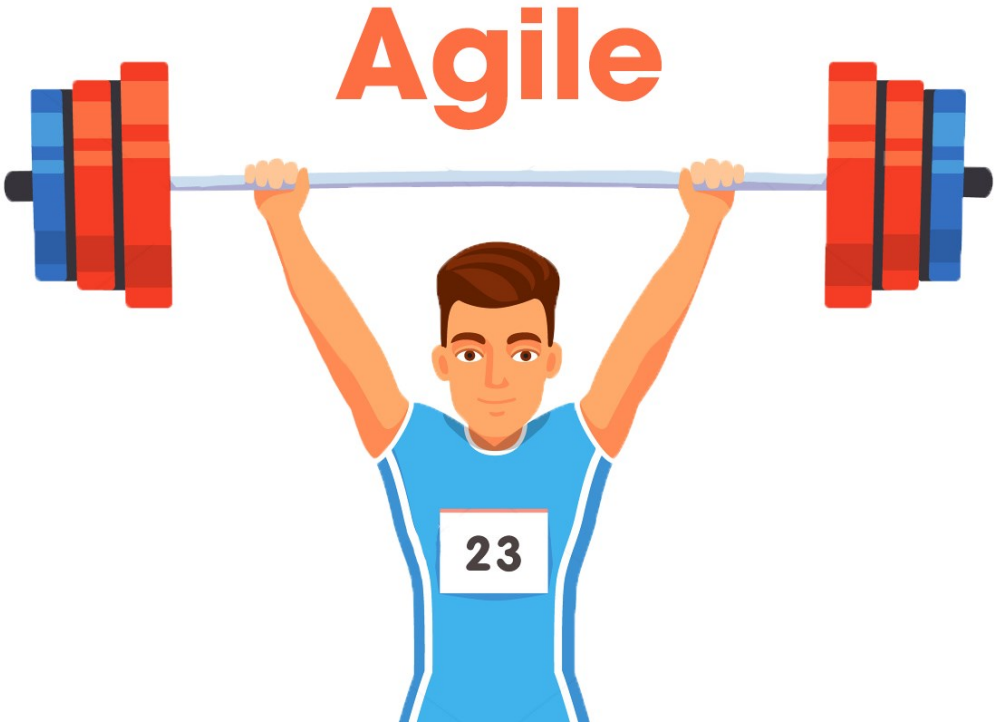
Regression is often only priority 1 or 2 tests where lower priority tests are of less importance, not automated, not re-run, so therefore not *all*.

Regression may only be the user story validation tests that were documented and automated along the way, based on how much time the automation folks had to automate whatever they did rather than focus on long lasting, confidence building, bug finding regression suites. The fact is, regression means many things to different people. This is particularly true where *automate everything* is a common mantra in continuous testing. There is a speed to continuous delivery and continuous deployment that leaves no room for *back-up plans*, the new description of regression has to be *lean and mean,* fast and furious and bug finding: not bloated, or the same old *whatever* we automated at the time.

For us to be testing more, and on more environments is pretty obvious. But the tests we are running have to change. Rerunning the same tests over and over on the same system with the same data against the same build does not help anyone. We know we have a smoke test suite—a build validation suite that is run in the CI process on the dev or test environment. We all have a full regression suite we normally run against the test environment. Here is where we need to begin reformulating.

For some teams, their automated full regression suite can take days to run. Many full regression suites have gotten bloated and slow.

- This is not OK. With the *Agile on Steroids* mentality, automation thought, theory, and practice needs to move from *bigger is better* to *lean and mean*. Continuous Testing suites need to be economical.
- Teams will need to redesign, optimize and even cut the size of their automated suites for Continuous Testing.
- You will need various suites of tests. You will have a smoke test (however you define smoke) and a bigger set of tests, probably called regression, which are already part of your CI process.



Will you run either of these in production? I hope not. The goals of those suites are not what you want in production! An automated suite of tests you run in production must be extra careful to maintain the integrity of the production environment, and users while monitoring system behavior and availability.

Will you run either of those suites against a big number of VMs? Do you need to? What tests will you run against each mobile device or emulator, for example? You need an intelligent and well-thought strategy for what tests you need to run, how often, against which environments.

Understanding the purpose of each environment and the monitoring or information needed from that environment is key to selecting which tests get run. For example, you may have five regression suites: 1) A fast smoke regression. 2) A big full regression. 3) A smaller *happy path,* end-to-end transactions/integration style regression suite, for the fully integrated environment. 4) A bug regression suite. 5) A main functionality—*lean and mean* regression suite. As well as a smaller set of tests—whatever you choose to run in production for monitoring—each will be run at various points in the DevOps process on various environments. It's important to remember, all this testing is in addition to the new feature development testing that happens inside your sprint. That testing continues as before.

All, meaningful, thoughtfully designed, prioritized tests—all regression tests—with different suite names.

Why do suites need to cut the bloat and be lean and mean? For a few reasons:

- Tests get old, and obsolete
- Tests need maintenance.
- Always be aware to cut poorly designed tests and failures. Tests that often fail or break easily need to be rethought and designed another way to find bugs or breaks faster and be less fragile.
- Get immediate feedback.
- An efficient set of tests will be run by other people on the team in continuous testing.
- Get the biggest coverage with the smallest number of tests. That is, get the *biggest bang for the buck*.

Testing in production

For all the years I have been in testing- it has always been a strict rule—**no testing in production**. The last thing you want is to have the testing messing around on a live system/environment with

customers only to have testing do something to muck up the system.

Only on rare occasions did the test team tread lightly onto live—to do something that could only be done on live or unique to a live customer environment. This has partly been changed with the use of virtualization. This has entirely changed in continuous testing.

You may be asked to run a suite of tests to validate the migration to live, or monitor the live system. Whatever the case, you need to make sure you are testing what needs to be tested and no more. Make sure the whole team knows the risks of testing in the live production environment and the risk of not

testing in the live environment, and as we have talked about your tests cannot interfere with the normal users workflow performance or operation of the system—be non-intrusive.

A note here before we begin talking about automation. Many people talk about automation in DevOps. Often, though, they do not mean test automation. They may be referring to task automation by the Ops team. Automating building environments, provisioning databases, populating user lists...these common tasks for many IT/Ops teams can now be scheduled, automated and run significantly faster due to the flood of new tools focused on Ops and the *Infrastructure as Code* movement. This is the flood of automation many people refer to.

Rethink automation

Continuous Testing relies on...continuous testing! Everything needs to be automated—or better, everything that can be automated, needs to be. This concept is the single key to successful continuous testing.

What tests need to be automated, what tests do not, how often they should be run and on which environments, how fast they are to run, the possibility the test will *fail*, the size of tests and their maintenance and other factors should impact what and how things will get automated. DevOps demands an entire rethinking of your automation program. For example, it is common to shrink your *full regression* to run faster and on more environments. Shrinking automation suites is an uncommon goal for many test teams.

Summary

The test strategy for continuous testing is a full reexamination of what gets tested where. This, with the Lean Software development (LSD) idea of "quality at every step" is the foundation of a continuous test strategy. It involves reexamining what you have been calling regression and developing multiple lean and mean suites for immediate feedback.

It also means rethinking what and how you automate. Old style automation—even small isolated automated tests need to be reexamined and replaced by efficient, low level tests as well as longer customer-focused transaction workflows. Rethinking regression and automation is a big task for even the most effective teams. This is a big job. Part 3 of this Continuous Testing in Dev Ops series deals with automation tool issues, environments and virtualization and cloud.

Checklist for Automation in Continuous Testing and DevOps

Automation Goals

⇒ The automated test suites need to be fast, efficient, effective.

⇒ The automation needs to be "same Sprint automation." Building up technical debt on test automation leads to failure.

⇒ A team may need to redesign, optimize or even cut the size of their automated suites for continuous testing.

⇒ Automation often becomes bigger and more complex but, needs to be faster in DevOps/Continuous Testing.

⇒ Remember suites need to be lean and mean.

⇒ The goal of DevOps is to give fast, immediate feedback on the system health with rapid delivery of new functionality, product stability and all the component parts and services on various environments.

⇒ You need sophisticated, thoughtful test automation for Continuous Testing. Simple, mindless automation will not benefit anyone.

⇒ Automation does not ensure quality—what automation does is measure consistency.

⇒ Your test automation skill, expertise and staff size should increase to satisfy the increased need.

⇒ Low maintenance framework is not a hope—it is the foundation of DevOps.

Automation Tests

⇒ Examine your tests and test design—be critical and honest about the value.

⇒ It has become common with many automation tools in use today to focus on small isolated validation tests. That is, tests have gotten atomic.

⇒ Small isolated tests do not simulate how your customers use the system. There is no real workflow.

⇒ Building small isolated tests is not as efficient as automated unit tests since UI driven are not at the code level, there can be other interactions between the code and UI causing problems.

⇒ We need automated tests that focus on full transactions, and end-to-end tasks and workflow.

⇒ Our tests exercise interaction, transactions, integration with a goal to find bugs and also validate through the UI.

⇒ This is especially true when your product uses services.

⇒ Even happy path tests are good to execute a path as long as they string together many functions.

⇒ Are your automated tests easily traceable?

⇒ Automation tests get old. Automation does have a shelf life. Have a good methodology and process to remove old tests as well as add.

TOP 10 MOBILE APP TESTING MISTAKES TO AVOID

By Daniel Knott

This article will cover 10 common mobile app testing mistakes to avoid when you are a software tester working in a mobile app testing and development environment. The 10 points may help you to start your mobile testing activities if you are new to mobile testing or they may help you to recap your existing mobile testing approaches.

1. Miss the platform UI/ UX guidelines

No matter if you are testing an Android, iOS or Windows phone app you as tester must know the different platform guidelines. Those guidelines include the interaction and the design as well as common development patterns. If you don't know them you can test your app against those requirements. However, every app must be aligned with the different platform guidelines.

2. Try to test everything

Well this is not only a pitfall for mobile apps but rather for every software out there. Testing everything is not possible and also not efficient. When you are new to an app start exploring the provided features and play with the app to learn about it. Try to narrow down the changes in the app and try to focus on those, e.g. use the context-driven testing approach to plan your testing activities. Use touring heuristics to be focused during your testing activities.

3. Mobile is not web

This point is especially hard for software testers who shift from web application software testing to mobile. The mobile use case is completely different than the web approach. Mobile users are on the move rather than in front of a stationary computer with a fixed location. Mobile apps are used in different environments e.g. during sport activities, while traveling or at work. Furthermore, mobile applications are optimized for smaller screens and have access to special hardware sensors. Therefore, it is very important for software testers to be on the move while testing the app and to keep the different mobile use cases as well as the different environments in mind.

4. Just look on the UI

Unfortunately, that is what many software testers are doing. Take a look at the UI if everything is looking nice. However, mobile apps are more than just a shiny UI. Mobile apps are using plenty of APIs and those APIs itself are worth checking. Is the API delivering the correct response to the requested call? How is the API handling wrong calls with wrong data? Besides checking APIs, mobile apps rely on many backend services. It is definitely worth checking the backend function too. Is the backend mature enough to handle slow requests coming from the mobile app in EDGE network? The network itself has a huge effect on mobile apps and therefore apps must be tested in different networks, too. Check also the integration with hardware features like the different sensors. There is many more stuff to test besides the UI.

5. No in the wild testing

As already mentioned in several blog posts and tweets, mobile apps are used by your customers while they are on their move in different mobile networks. Mobile testers must leave the office to test the app in real life environments to catch some bugs that may only occur in the different network conditions.

6. Try to automate as much as you can

Writing automated checks for software is not easy. Writing automated checks for mobile apps is even more complex given all the dependencies to APIs, networks, sensors and backend systems. 100% test automation is not possible and also not efficient! Writing automated checks is expensive if you keep in mind that those checks need developing and maintenance time. Automated checks are also nothing that should be done, just because there must be some checks in place. Automated checks must be treated as production code and this requires that skilled and educated people should work on that topic. Have the mobile test pyramid in mind when you want to start test automation for your mobile apps.

7. Test only one Single Device

As we all know the mobile market is highly fragmented when it comes to hardware and

software combinations. This is not an Android problem anymore, the same problem exists on all mobile platforms. Therefore, it is very important to test on various mobile devices to have a higher coverage of the different devices. However, it makes no sense to test on Android or iOS devices that are not used by your customers. Concentrate on those devices which are used by your target audience.

8. Don't Listen to your Customers

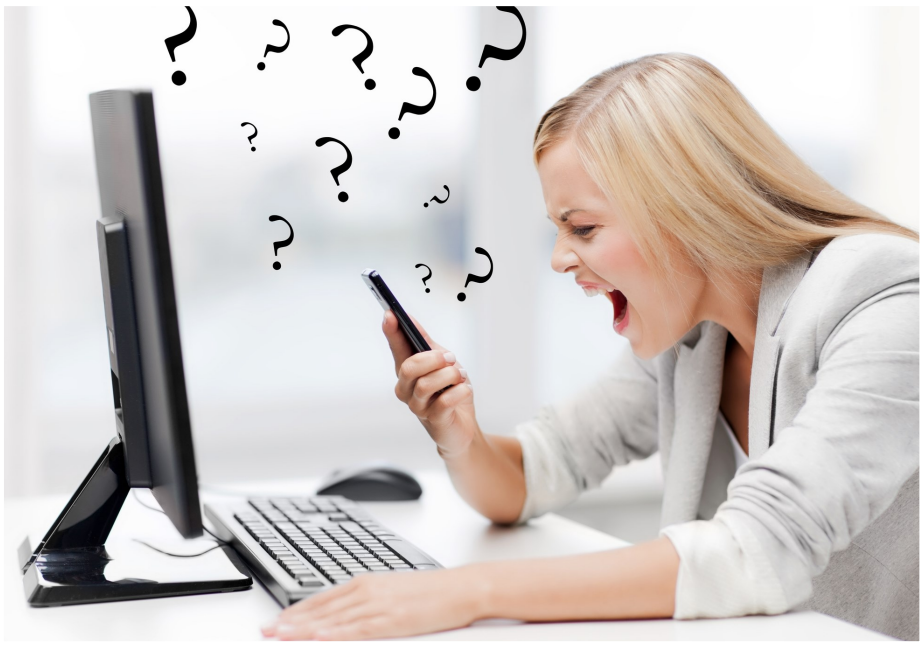
Check the app store reviews of the mobile platforms on a regular basis to see what your customers say about your app. Sometimes there is valuable feedback that may help you to find and fix a problem. It is also worth to check the social media platforms for error, bug reports or direct feedback that indicate problems. If the company has a customer support department, go to your colleagues and talk to them about possible problems. If you have the chance to talk or respond to your customers use that instrument to get in contact with them. In that way customers feel appreciated and welcome and they may change their mind about the product and will provide better feedback once the problem is solved. If there are customers who deliver great feedback ask them to become a beta tester of your app to use their engagement in the product to turn them into a valuable source.

9. No update/ installation testing

Installation and update testing an app is crucial for every mobile tester. The installation is the first thing a customer will see from your product and from your company. If the first impression is a bad one, they will most likely delete the app with a single tap from their devices. Therefore, it is important to test the installation and update process of your app. Read about update testing in this post.

10. No security testing

Most of us know that security is a very important topic for every software product out there. We also know that security testing is not easy and therefore it is really important to talk to experts to ask them in security related topics. A first starting point for all of you might be the OWASP pages, they offer nice lists and overviews of potential security problems in software. However, use those lists to question your app and the architecture the app relies on. Then get in contact with experts to check mobile apps for security problems. Sad but true, security testing comes in many cases way too late when really bad stuff happened.



Now that you have read the 10 common mobile app testing mistakes, you should check your testing activities if you miss something in your daily work. Maybe the overview will help you to become a better mobile tester.

#HappyTesting

Daniel Knott

Daniel is a mobile testing expert working as Senior Software Test Engineer at XING's Android team. He started his software testing career in 2003 as a trainee at IBM Germany. After his time at IBM, Daniel studied computer science with a focus on software development and testing. Since 2009, Daniel has worked for companies such as Accenture, AOE and XING. In several agile development projects he tested web, desktop or mobile applications. However, mobile testing became his passion and since the beginning of 2011 he is working in the mobile development and testing industry. He worked and is working with several mobile test automation tools such as Robotium, Calabash for iOS/ Android, Espresso and Keep It Functional. With the help of this tools, he developed a fully automated testing environment for Android and iOS.

Daniel likes to share his knowledge and therefore he started to share his experience on his blog www.adventuresinqa.com as well as in several testing magazines. Daniel is a well-known mobile expert, a speaker at various conferences in Europe and since 2014, he is the author of the book "Hands-On Mobile App Testing", more information about the book can be found at www.handsonmobileapptesting.com.

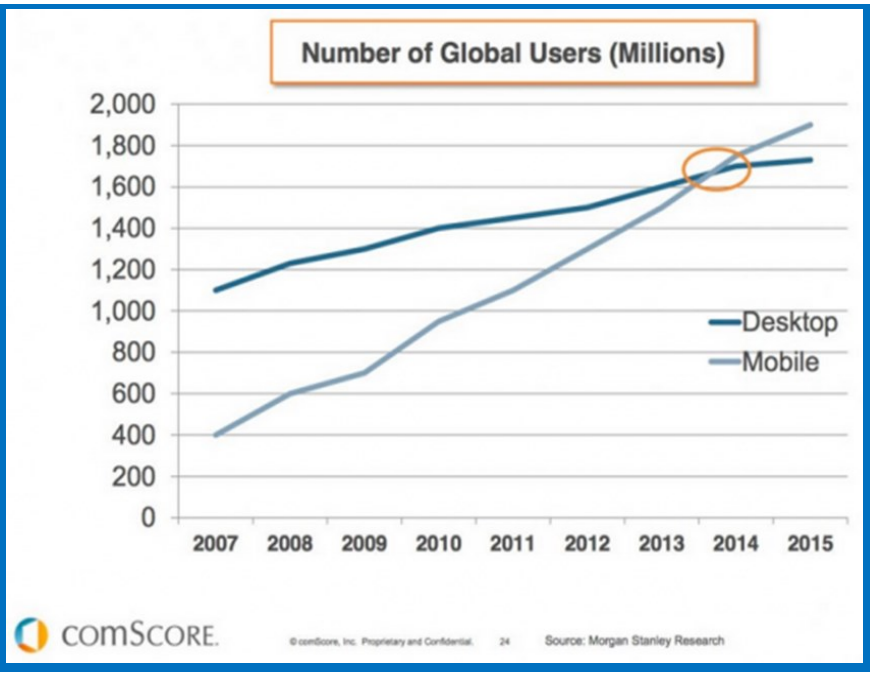
Get in touch with Daniel:

- Twitter: @dnlkntt (https://twitter.com/dnlkntt)
- XING: Profile (www.xing.com/profile/Daniel_Knott)

LEVERAGE MOBILE TESTING WITH TESTARCHITECT

BY VAN PHAM

Mobile usage today is not just a trend but it is an essential shift in how people communicate with each other, interact with the world, and do business. According to a ComScore, in 2014 the number of mobile users surpassed the number of computer users and is showing strong growth over time, towards some point in the near future where virtually each person on Earth will own at least one mobile device.



Therefore, it is no longer a case of asking whether building mobile apps or mobile compatible websites is essential for any business. It is. "Mobile first" has become a motto for many companies nowadays to keep up with the race of mobile age, in order to retain customer loyalty.

Users quickly uninstall a mobile app if they encounter an application bug, slow loading speed or incompatible issues with their devices, so making sure the application works seamlessly on any device is vital. On the other hand, developing and testing on mobile devices are quickly getting overwhelming because of the variety of operating systems and their versions, of device makers and models, and of screen sizes and resolutions. And that doesn't even cover the various input methods, carriers, data transference speed, GPS, accelerometers, and application kinds: native, web or hybrid apps.

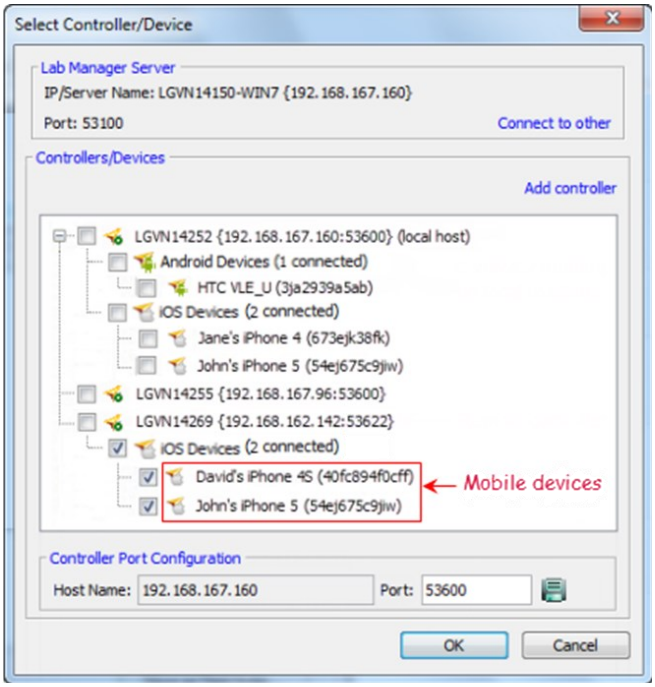
Add to this the extensive infrastructure of backend systems to support the mobile applications and their usage. Clearly, testing plays an important role in ensuring that requirements are met by ensuring apps run well and consistently on a set of different target devices. However, creating and managing a large amount of test cases which cover all combinations of mentioned variances in a short of time is not an easy job.

The TestArchitect automation tool relieves these concerns in an elegant way by applying Action Based Testing methodology into test design to quickly create tests. No programming knowledge is required

and maintenance effort is kept at a minimum. For the test execution, TestArchitect allows user control via cable and/or Wi-Fi from one host machine up to 20 devices, that can also include simulators. A user can use the screenshot capturing feature to record the run on each device.

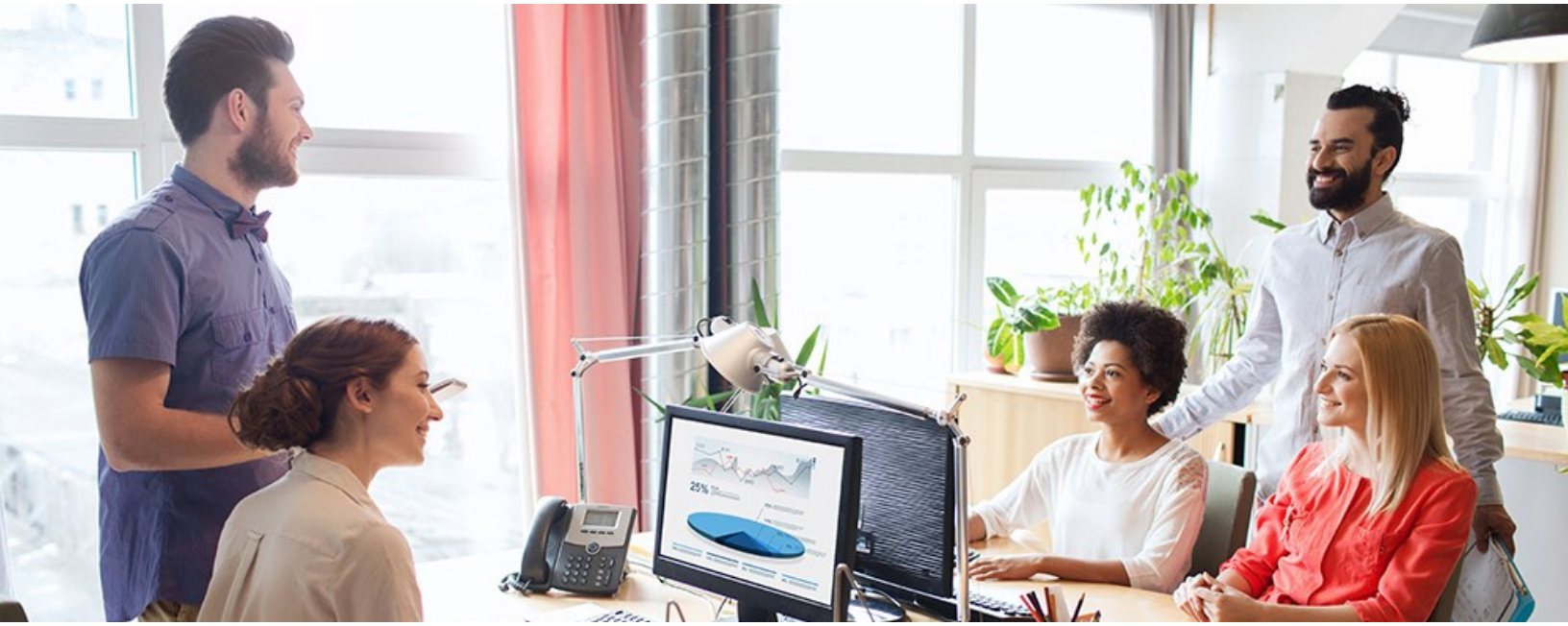
When the execution completes, the test results of each device will be collected by the host machine. From the test results, QA staff can see if the application under test has issues on specific devices and work with the development team to fix the issues before releasing the app to the market.

In addition to the mobile support, TestArchitect supports a wide range of platforms like desktop, web, and non-UI, for example database access and REST service calls. This allows you to flexibly combine device level testing with related testing on the backend systems.



GET YOUR TEAMS THE TRAINING THAT THEY NEED!

BY MICHAEL HACKETT



Every leader knows having a fully trained staff is essential to productivity, employee satisfaction, teamwork, and just getting the job done!

This is particularly true today in technology for many reasons.

I want to focus on three ideas.

1.Training for training's sake:

Training to keep up with tech changes.

Training as a tool for better employees and employee satisfaction.

2.Training on software development, process, communication, collaboration.

3.Training for new technology.

1 Training for training's sake. Technological advances are happening at a significantly rapid pace. Just keeping a team's skill-set to get the job done correctly is a headache for any manager. When the product or platform grows teams must be ready to hit the ground running.

Training to keep up with tech changes. I think of the time when agile was first being introduced into organizations and the direction from development managers was: cut release time, stop documenting, we're going to move your seats around, the project lead is now called the scrum master, and now we're going to be *agile*. Most times those changes were made without the needed work culture change. They were made without the benefits being communicated to the staff: namely being that they were going to lead to a better customer experience and shorter release times. As a result many teams adopted *Scrumbutt* practices to meet these demands. It took some organizations years to catch up to implementing agile and scrum right. I saw a

great lack of leadership in many organizations that implemented scrum wrong. This could easily have been remedied or prevented by giving a few staff workshops in trainings from qualified people to have everybody understanding what these culture and practice changes were about.

Training as a tool for better employees and employee satisfaction. I like to refer to this as continuous learning. Technology employees are in great demand. Working in a continuous learning environment is essential to staff retention. It is often a hiring criterion for prospective employees. What is the annual training budget? Who manages my career path? How often do you bring in outside training? These are all commonly asked questions.

Every manager knows that having a skilled staff is essential to getting the job done. But how much is a manager responsible for their staff training? This is an age old question. Having worked in Silicon Valley for my whole career two answers come to mind. As technologies change, it impacts changes to both staff and products. It's important that your team knows how to do their job—that is a leader's job. It is your responsibility to make sure that your staff is fully equipped with all the knowledge they need to get the job done. Having said that, many companies in Silicon Valley rely on the individual to keep their skills up to date. Most companies do not provide a career path for their employees. This is unique to the Silicon Valley mindset where knowledge is all powerful. It is the individual's job to make sure they can compete in the workforce and get the job done. On the other hand, many companies use training for a staff retention tool. For leaders, this means balancing the individual's job to direct their career with creating an environment where learning is valued, appreciated and paid for, sometimes to attract the best employees. It is a long known fact that many knowledge workers in technology will apply for a job or stay in a current job because they're learning cool technologies or using cool technologies, even if they are paid less, compared to if they were to stay or join a company that would be using older or not as cool technologies and platforms.

2 Training on software development, process, communication, collaboration—just when some teams are feeling more comfortable in agile development many organizations are now moving past Agile, or *graduating* from Agile to Kanban development, or Lean Software Development (LSD). This change to Kanban has been particularly rapid when dealing with customer support and production issues. Kanban is not the same as scrum and a little bit of training can go a long way in helping teams reduce stress and maximize their performance. Many organizations are moving from regular interval releases using agile development to continuous delivery or continuous deployment using DevOps practices. Product development teams really need to know what they are talking about. For example, Continuous Testing is ***not*** merely testing continuously.

Currently I see a huge need for training concerning the move to DevOps. This is a real turning point in software development where we need culture shifts, team building, and technical knowledge before any significant changes should be attempted. Having operations people, programmers and testers working daily with business side people may not go as smoothly as some companies wish. These teams do not have a long history of working together on the same projects, but it can be successful as long as there are culture shifts, in addition to work habit shifts. What DevOps is and how it will be implemented across organizations or even pilot projects requires rethinking roles and responsibilities. Although DevOps will challenge operations people more than other people on the team, testers in particular have a new way of working with far greater responsibility in continuous testing. This will directly impact test environments, regression testing and test automation. The demand for training is huge.

3 **Training for new technology**—many companies are familiar with mobile test strategies, but they now have the SMAC stack to contend with. SMAC—Social, mobile, analytical, and cloud technologies all rolled up into one development platform.

The rapid growth of mobile testing and mobile platform products took many people by surprise. That platform has now evolved into a much more technically complex set of technologies called the SMAC stack. Testers need significant technical training to be able to handle this shift. Many testers who had been just getting on their feet with mobile testing or even web service and social API testing are now being asked to do this in the cloud with cloud tools. A whole new area of analytics also has to be tested. You can read from many sources how analytics is impacting product development and design as well as modeling the customer experience. Test teams are in the middle of the analytics world in their work.

This is a brand new arena for many test teams. The first thing we all need to agree on is that it is a manager's job, to make sure the team has the skills to do their job effectively.

The question for leaders today is how you'll ensure that your staff gets those skills.

There are great traditional practices to getting your staff's skills up. You can do brown bag lunches where you give trainings for an hour, during lunch time, one day a week. These tend to be very effective as hour chunks of training, and are not as difficult as a full-day training to develop. Also, an hour at lunch is a manageable amount of time, and can still have great impact. You can find some good videos on just about any technology topic and have the team watch them individually, or as a team. To reinforce the learning and make sure all the staff is participating you can have a group discussion afterwards. Doing so enables you to have a minimum amount of content development time.

You can find articles and case studies on all of these new technologies, pass them out to the team to read, and follow the same process or examples as the videos and have group discussions afterward.

The most effective method, which also may be the most traditional, is where you bring an outside trainer in to hold a workshop or teach various topics. There are a lot of hidden values to this style of learning. When the training becomes an event it tends to be given more weight and focus then having individuals read an article or watch a video. By having a forum where teams can ask questions, argue and discuss, it tends to get everyone on the same page and reinforce the learning. Also bringing in an outside expert on the technologies you need is a quick way to get high value learning to a big group of people. As a leader this is one of your core functions: continuous learning for you and continuous learning for your staff! How you do it is up to you. Good luck!

8 LESSONS LEARNED IN MOBILE APP TESTING

BY ALISTER

I've spent the last six months or so testing mobile apps for both iOS and Android. Here's eight of my key lessons learned:

1. Automated UI testing tools for mobile apps are immature: whilst tools like WebDriver for automated UI testing of web apps are very mature, automated UI testing of native mobile apps is the opposite. Whilst Appium shows some promise as a cross-platform UI testing tool for mobile apps, in my opinion, the maintenance overhead is still high as the tests are quite flaky. I recommend more thorough automated unit and integration testing, possibly supplemented with WebView tests (see below) and some exploratory manual app testing of the UI.

2. Base your mobile OS testing on *projected* usage, not current usage. Mobile hardware and OS upgrade cycles are shorter than desktop hardware, and if you look at trends and test against those, chances are you will have more realistic usage based testing.

This chart from mixpanel shows iOS7 adoption across a very large number of devices. You can see iOS6 is trending downwards as iOS7 is broadly adopted, so you could put less focus on testing iOS6.

3. Test on real devices: I prefer to test on real devices like phones and tablets so that I can get a realistic feel for how the app runs. There's also some things that can't be tested on emulators/simulators like push notifications which require a device id, and Apple VoiceOver for accessibility testing which isn't available in the simulators.

I have found that iPod Touch's make excellent test iOS devices as they're a lot cheaper than iPhones, and you can buy them refurbished on the Apple store, although if you want to test in the wild (see below), you'll need a cellular iOS device. If you're after an iOS6 specific test device you're only option is to buy a 4th generation iPod touch refurbished as all new Apple iOS devices now come with iOS7 pre-installed.

It's easy to find cheap Android pre-paid phones for testing, and you can still buy Android 2.3 phones with lower-res screens which is good for older version compatibility testing.

4. Enable wireless distribution/installation of your test application locally: to enable quick installation of updates to your app on both iOS and Android.

For wireless **iOS** distribution this involves have a build machine (which has to be a Mac) compile an IPA file which is copied to a web-server, along with an XML manifest (PLIST) file that describes the app. You then have a simple web page with a *itms-services* link to the manifest, which can be opened in Safari on iOS, prompting the user to automatically install. If you sign your iOS app using an *enterprise* account it will run on an unlimited number of iOS devices without registering them as test devices.

For wireless **Android** distribution it is much easier. Simply have your build server compile an APK file and copy it to a web-server along with a page linking to the APK directly. Any Android phone set to allow installation of apps outside the Play Store will download the file and prompt to install it. *Voila.*

5. Test in the wild: I like testing during my bus/train ride home to see how my app handles with intermittent cellular network coverage, especially in underground tunnels. If your backend services aren't publically available you can use a VPN client on your phone to allow you to test against local infrastructure.

The one thing you need to be aware of is other people on your bus/train stickybeaking if your app is commercially sensitive.

6. Test WebView/bridged app HTML content independently of your app: we use a service that returns HTML5 content that is displayed and executed using a JavaScript bridge in both iOS and Android native apps. This content can be tested independently of the native apps which is both faster and easier than testing via the apps themselves.

7. Fake back-end dependencies for quicker UI testing. Back-end dependencies of your application such as a database can be faked so you can quickly test your app's UI is functioning as designed initially without having to worry about setting up an entire application stack for your testing.

8. Test for accessibility on real devices: using iOS VoiceOver & Android TalkBack. These are both excellent screen readers which use very similar gestures to control the operating system and apps on your phone. The most common accessibility issues I discover are missing accessibility labels for UI controls and embedded WebViews not being marked accessible to the screen reader tools.

Summary

There we have it: eight tips for testing mobile apps. What lessons have you learned in the mobile testing space?

Alister

Alister is an Excellence Wrangler for WordPress.com at Automatic. He has extensive experience in automated software testing and establishing quality engineering cultures in lean cross-functional software development teams. He lives in Brisbane, Australia with his wife and three sons, and writes a popular software testing blog at watirmelon.com.

SMAC: SMAC creates an ecosystem that allows a business to improve its operations and get closer to the customer with minimal overhead and maximum reach. The proliferation of structured and unstructured data that is being created by mobile devices, sensors, social media, loyalty card programs and website browsing is creating new business models built upon customer-generated data. None of the four technologies can be an afterthought because it's the synergy created by social, mobile, analytics and cloud working together that creates a "competitive advantage.

Source: http://searchcio.techtarget.com/definition/SMAC-social-mobile-analytics-and-cloud

Social Media: Social media has provided businesses with new ways to reach and interact with customers, while mobile technologies have changed the way people communicate, shop and work. Analytics allow businesses to understand how, when and where people consume certain goods and services and cloud computing provides a new way to access technology and the data a business needs to quickly respond to changing markets and solve business problems.

Source: http://searchcio.techtarget.com/definition/SMAC-social-mobile-analytics-and-cloud

Web services (WS): Web services (sometimes called *application services*) are services (usually including some combination of programming and data, but possibly including human resources as well) that are made available from a business's Web server for Web users or other Web-connected programs.

Source: http://searchsoa.techtarget.com/definition/Web-services

API: *Application program interface* (API) is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact and APIs are used when programming graphical user interface (GUI) components.

Source: http://www.webopedia.com/TERM/A/API.html

IoT (the Internet of Things): The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

Source: http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

Microservices: Microservices are a more concrete and modern interpretation of service-oriented architectures (SOA) used to build distributed software systems. Like in SOA, services in a microservice architecture[1] are processes that communicate with each other over the network in order to fulfill a goal. Also, like in SOA, these services use technology agnostic protocols.[2] Microservices architectural style is a first realization of SOA that has happened after the introduction of DevOps and this is becoming the standard for building continuously deployed systems"

Source: https://en.wikipedia.org/wiki/Microservices

Security testing: Security testing is a process intended to reveal flaws in the security mechanisms of an information system that protect data and maintain functionality as intended.

Source: https://en.wikipedia.org/wiki/Security_testing

Service-oriented architecture (SOA): Service-oriented architecture (SOA) is an approach used to create an architecture based upon the use of services. Services (such as RESTful Web services) carry out some small function, such as producing data, validating a customer, or providing simple analytical services.

Source: http://searchsoa.techtarget.com/definition/service-oriented-architecture

Vulnerability Scanning: Vulnerability scanning is a security technique used to identify security weaknesses in a computer system. Vulnerability scanning can be used by individuals or network administrators for security purposes, or it can be used by hackers attempting to gain unauthorized access to computer systems.

Source: https://www.techopedia.com/definition/4160/vulnerability-scanning

Penetration Testing: A penetration test, informally pen test, is an attack on a computer system that looks for security weaknesses, potentially gaining access to the computer's features and data.

Source: https://en.wikipedia.org/wiki/Penetration_test

Data Warehouse: In computing, a data warehouse (DW or DWH), also known as an enterprise data warehouse (EDW), is a system used for reporting and data analysis, and is considered as a core component of business intelligence[1] environment. DWs are central repositories of integrated data from one or more disparate sources. They store current and historical data and are used for creating analytical reports for knowledge workers throughout the enterprise.

Source: https://en.wikipedia.org/wiki/Data_warehouse

BI Testing: Business intelligence can be defined as a collection of approaches for gathering, storing, analyzing and providing access to data that helps users to gain insights and make better fact-based business decisions.

Source: http://ap-institute.com/big-data-articles/what-is-business-intelligence-(bi).aspx

EVENTS

STPCon-Fall

September 19th - 22nd, 2016

Hans Buwalda will be giving his presentation *Using Keywords to Support Behavior Driven Development (BDD)* and leading the workshop *What Makes Automated Testing Successful?* Michael Hackett will be leading two workshops at STPCon-Fall this year: *Move into DevOps: Experiences from the real world*, and *Test Case Design Intensive*. The Software Test Professionals Conference is the leading event where test leadership, management and strategy converge. The hottest topics in the industry are covered including agile testing, performance testing, test automation, mobile application testing, and test team leadership and management.

STARWEST Software Testing Conference

Oct 2nd - 7th, 2016
Anaheim, CA

Hans Buwalda will be presenting two testing tutorials this year, and LogiGear will also be exhibiting in the Expo. STAR*WEST* is one of the longest-running, and most respected conferences on software testing and quality assurance. The event week features over 100 learning and networking opportunities and covers a wide variety of some of the most in-demand topics.

STARCANADA

October 23rd – 28th, 2016
Toronto, Ontario

Hans Buwalda, will be presenting a tutorial Tuesday on Test Design for Better Test Automation. STARCANADA is part of the STAR family of conferences. IT is one of the longest-running and most respected conferences on software testing and quality assurance. The event week features over 50 learning and networking opportunities and covers a wide variety of some of the most in-demand topics.

All Things Open

October 26th – 27th, 2016
Raleigh, NC

Hans Buwalda will be giving his presentation Using Keywords to Support Behavior Driven Development (BDD). All Things Open is a conference exploring open source, open tech, and the open web in the enterprise. The conference is in partnership with opensource.com and leading technology and business leaders will participate.

LogiGear

OVER

LOGIGEAR MAGAZINE

SEPTEMBER 2016 | VOL X | ISSUE 3

United States
4100 E 3rd Ave., Suite 150
Foster City, CA 94403
Tel +1650 572 1400
Fax +1650 572 2822

Vietnam, Da Nang City
VNPT Tower, F/17 & 8
346 Street 2/9
Hai Chau District
Tel: +84 511 3655 333

Vietnam, Ho Chi Minh City
1A Phan Xich Long, Ward 2
Phu Nhuan District
Tel +84 8 3995 4072
Fax +84 8 3995 4076