Agile is All About Change...

So Why Does Testing Remain the Same?

By Wolfgang Platz, Founder and CPO

TRICENTIS

tricentis.com

The More Things Change...

Agile is all about change. IT leaders adopt Agile to accelerate the pace of change for their business-differentiable software. The adoption of Agile requires changes in the people, processes, and technologies involved in building that software. Development teams must significantly change their structure, culture, tooling, and daily activities to accommodate Agile. And once Agile is adopted, the applications under development change on a daily (or more frequent) basis.

In spite of all this change, one thing tends to remain the same: the software testing process. One recent study reports that 70% of organizations have adopted Agile, but only 30% automate testing. A separate study found that while Agile adoption is now near 88%, only 26% of Agile organizations have broadly adopted test automation. In other words, testing processes remain stuck in the past even as organizations invest considerable time and effort into transforming their development processes to meet today's and tomorrow's business demands. Not surprisingly, that same study reports that the majority of these Agile teams are not satisfied with the current pace of change.

Why does testing lag behind? In most cases, teams want to avoid the perceived pain of transitioning from a manual testing process to an automated one. If the acceleration initiative doesn't require a change to the testing process, testers typically won't take it upon themselves to initiate process change. Even if change is mandated, test managers tend to reassure the organization that adding some UI-level test automation to their existing test process will be sufficient.

Any attempt at test automation is definitely a step in the right direction. However, more is required to meet the needs of modern development processes. Forrester reports:

"Functional testing is one of the most crucial, time-consuming, and expensive steps in continuous testing—so it's necessary to automate this testing, and to automate it at higher levels than most Agile teams achieve today... To keep up with the pace of the rest of the pipeline, functional testing needs to be automated and optimized from beginning to end, from the design and automation of test cases, to their execution in the overall testing process, all integrated in within the broader CI/CD automation process."

To simplify:

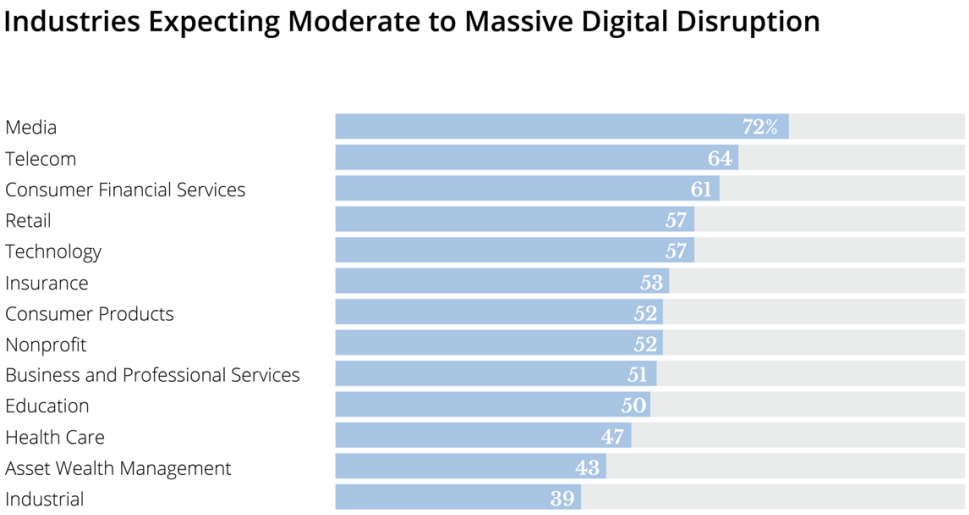
Continuous Testing > Test Automation.

In addition to changes in development, DevOps and Agile requires Continuous Testing. And for true Continuous Testing, you can't simply take a fundamentally manual quality process and add some automation. As many IT leaders I've worked with have found, it either increases your risk of ending up in the "software failure" headlines, or it creates a quality bottleneck that prevents you from achieving the desired speed. It's like Amazon investing considerable resources in optimizing every aspect of their ordering and fulfillment process, but still relying on ground shipping for expedited delivery.

In this paper, I want to explore why and how organizations must evolve their quality processes to fully realize the benefits of modernization efforts such as Agile and DevOps.

Digital Disruption is Driving Agile and DevOps Adoption

From financial to healthcare and retail to construction, no industry today is immune to digital disruption.



In order to be disrupting (rather than disrupted), the trend is for organizations to adopt initiatives such as Agile and DevOps to accelerate time-to-market. The latest industry studies indicate that 88% of organizations have adopted Agile and 80% have adopted DevOps.[1,2]

[1] Testing Trends in 2016: A Survey of Software Professionals, Sauce Labs: http://cdn.agilitycms.com/sauce-labs/white-papers/sauce-labs-state-of-testing-2016.pdf

[2] RightScale 2016 State of the Cloud Report: DevOps Trends, RightScale:digit http://www.rightscale.com/lp/devops-trends-report

Change is Intrinsic to Agile

Today's DevOps pipelines are typically fed by continuously-evolving software developed according to Agile methodologies. In fact, it is widely accepted that the increased number and frequency of releases that stemmed from Agile adoption is what drove the need to reliably deliver software faster and more frequently —the impetus for DevOps.

As I stated earlier, any Agile adoption is always fraught with change. To start, you need to:

- Arrange scrum training for all impacted development staff
- Designate a scrum master for each team and provide that person additional training
- Hire Agile coaches to help the team transition into the new process and culture

Next, you need to provide the appropriate tools for planning, tracking, and managing sprints—and organize yet more coaching to ensure that those tools are properly configured and that the team is using them effectively within their new process.

Once the adoption is in-flight, the daily development activities must change in terms of:

- Speed- The team adopts short iterations (sprints)
- Communication- The team holds daily stand up meetings
- Ownership- Product owners become embedded in the process
- Feedback- Sprint reviews and retrospectives provide feedback to fuel a continuous improvement loop
- Transparency- Burn-down and burn-up tracking quantifies and communicates progress

Given all the change required to just get such an initiative off the ground—much less to see it succeed—executive buy in and sponsorship is key. I've found that most IT leaders do recognize the value of Agile, and are committed to driving its success throughout the development organization. I firmly believe that extending this commitment to the testing process as well as the development process is key for fully realizing the ROI of the Agile adoption, and preparing the organization to face the next wave of digital disruption.

Transforming Testing for Agile and DevOps

Continuous Testing > Test Automation

Testers across the industry have already recognized that testing must adapt in order for testers to remain relevant in Agile and DevOps processes. In a recent survey of software testing professionals (including testers and managers):

- 91% believe that the role of the tester must be transformed to meet the needs of today's software development processes
- 70% believe that as organizations embrace Agile, the role of a tester changes significantly
- 45% increased the level of functional test automation during the past year in response to Agile

Moreover, Gartner confirms that as organizations accelerate software delivery, "testing simply cannot keep pace [with demand for faster delivery] due to a heavy reliance on manual testing processes." Additionally, survey after survey is now citing inadequate test automation as a principal barrier towards Agile adoption.

The writing is on the wall: manual software testing must evolve in response to the shift to Agile and DevOps. No matter how many testers you employ, it's simply not possible for manual testing to provide Agile developers immediate feedback on whether any of their constant changes impacted the existing user experience. Without this safety net, Agile becomes a tremendous business risk.

As the industry has begun to recognize, test automation will play a critical role in controlling that risk in an accelerated delivery process. However, not all test automation is equally effective for this purpose. When manual testers adopt automation, the most common path is to focus on UI test automation. However, most automated UI testing is not perfectly suited to the type of Continuous Testing that Agile and DevOps require because:

- Testing begins late in each sprint: Tests usually cannot be implemented until late in each sprint—when the UI and dependent components such as back-end APIs are finally completed and available for testing.
- Slow execution time: Tests are time-consuming to execute, so it is not practical to run the complete regression test suite on each build. This means the team lacks instant feedback on whether their changes impact the existing user experience.

- **High maintenance:** UI tests require considerable rework to keep pace with the frequent changes typical of accelerated release processes. This results in slow, burdensome maintenance and/or causes automation efforts to be abandoned.
- **Frequent failure:** Test environment inconsistencies (e.g., intentionally dynamic elements or frequently-changing dependencies) can also cause fragile UI tests to fail—again, requiring burdensome follow-up and/or causing automation efforts to be abandoned.

Automated UI testing certainly is not dead. However, like everything else, it too needs to be re-engineered for Agile and DevOps.

Based on our experience helping manual testers at leading organizations transition to Continuous Testing, we've developed the following guidelines for changing testing to advance Agile and DevOps processes:

#1: Prioritize Automated API Testing

Most software testing professionals are familiar with the Agile test pyramid, which de-emphasizes the system integration tests and user acceptance tests (a staple of manual testing for years) in favor of increased emphasis on integration tests, which focus on the API layer.

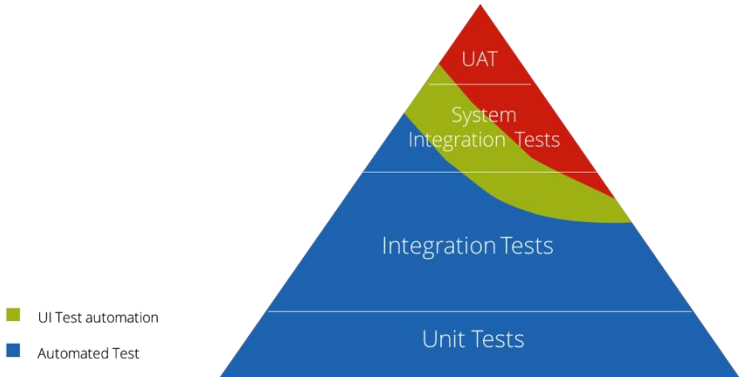
API testing is widely recognized as being much more suitable for Agile because:

- Since APIs (the "transaction layer") are considered the most stable interface to the system under test, API layer tests are less brittle and easier to maintain than UI tests.
- API tests can be implemented and executed earlier in each sprint than UI tests.
- API tests can often verify detailed "under-the-hood" functionality that lies beyond the scope of UI tests.
- API tests are much faster to execute and are thus suitable for checking whether each new build impacts the existing user experience.

In fact, our recent studies have quantified some of the key advantages of using API testing versus UI test automation:

Task	UI test automation	API testing	Factor
Set-up	100%	25%	4x
Maintenance	100%	16%	6x
Runtime	100%	<1%	100+ x
Timing	regressive	*progressive*	

This leads us to our recommended take on the test pyramid:



The red tip of the pyramid indicates the role that manual testing (typically via exploratory testing) is best suited to play in modern development processes. The green band represents what we've found to be the "sweet spot" for UI test automation. The vast majority of the triangle is covered by API testing, which builds upon the solid foundation of development-level unit testing.

#2: Use Service Virtualization and Test Data Management to Help Teams "Test Early and Often"

Executing a test that realistically assesses the end-user experience requires access to a complete test environment—including all the systems that the application under test interacts with, as well as realistic test data. However, over half of the average application's 50+ required dependencies can't be reliably accessed for testing because they are inaccessible, evolving, or not yet implemented (which is extremely common with Agile/parallel development).[3] Furthermore, getting a test environment configured with appropriate test data can take weeks (and a set of test data is often applicable for only a single test run). Given

[3] voke Market Snapshot: Service Virtualization 2015, Theresa Lanowitz and Lisa Dronzek: https://www.vokeinc.com/enterprise-service-virtualization-data.html

these challenges, how do you test each user story as it's completed as well as continuously test each new build?

Many leading organizations have found that simulation helps them create a complete and consistent environment that is perfect for continuous test execution. Service virtualization simulates interactions with dependencies that are not available or configurable for testing, and test data management instantly creates and applies test data appropriate for each and every test. In addition to enabling early and continuous testing, these technologies also facilitate defect reproduction—helping teams achieve the Agile goal of fixing defects as soon as they are discovered.

#3: Optimize the Test Portfolio to Increase Risk Coverage While Reducing Maintenance and Execution Time

The most mature Agile and DevOps teams often use a functional test suite as a "quality gate" to stop high-risk release candidates from progressing through the software delivery pipeline. This requires the testers to design a test portfolio that accurately exposes top risks—but is lightweight enough that it doesn't clog the delivery pipeline. This is a tall order in any context, but it becomes even more daunting when you consider the extremely limited time available for testers to design and implement tests in each Agile sprint.

With risk-coverage optimization working hand-in-hand with model-based test automation, testers can rapidly identify and create an optimal test portfolio that focuses on the organization's most important business risks. Teams working with such technologies have been able to reduce the number of test cases by 50%-80% while increasing risk coverage to up to 90%. The result is an extremely efficient quality gate that identifies and contains business-critical risks—elevating the role of testers within the organization while actually reducing their test design and maintenance burden.

Wrap-Up

When you adopt Agile development, you accept that you will need to change your development processes and infrastructure in a number of significant ways. Unless you are also willing to accept that Agile requires a testing change towards Continuous Testing—which involves optimizing and expanding upon traditional UI test automation—there's a high risk that your results will fall short of expectations. Even the most basic test automation initiatives will quickly reach a standstill if the team lacks the required test environments and data. To maximize the value of your organization's investment in Agile, you can't simply automate testing; you need to do it in a way that suits the needs of today's highly-complex and distributed enterprise applications. In addition to flexible UI-level test automation, this also requires risk coverage optimization, API-level testing, service virtualization, and test data management.

Want to try Tricentis Tosca in your own environment?

Start your free trial of Tricentis Tosca today.

About Tricentis

Tricentis, the Continuous Testing Company, specializes in market leading agile software testing tools for enterprises. We help Global 2000 companies adopt DevOps and gain success by achieving automation rates of over 90%. Top analysts recognize Tricentis as a leader in Software Test Automation, with Model-based Test Automation and Test Case Design as standout features.

Our 400+ customers include global names from the Top 500 brands such as A&E, Allianz, Deutsche Bank, HBO, JetBlue, Orange, Swiss Re, Telstra, Toyota, UBS, Vantiv, Virgin Airlines, and Zurich Insurance. Tricentis has offices in Austria, United States, Germany, Switzerland, UK, Netherlands, Poland, and Australia.