TESTCo
Outsourced Software Testing
www.testco.com

Is Your Agile Software Testing Team Wasting Money on Automation Tests?

Four reasons why your customers are finding defects that your agile software testing team's automation tests didn't catch

Overview
Agile software development teams typically rely heavily on the agile software testing team's ability to automate regression testing to ensure quality in their software. For some agile methodologies, automation is actually part of the methodology (i.e. XP and Lean Product Development), while in others it is implied or simply necessary to meet the goals of the methodology.

The problem is, developers and most software testers aren't trained to create automated regression tests. They are trained to create unit tests or functional tests. What we often find are suites of test automation scripts that are great for the developers, but almost completely useless when it comes to functional or system level regression testing.

This report examines the challenges and pitfalls often involved in creating automation tests in an agile environment.

1. Agile development requires the existence of automated regression testing to reliably deliver shippable code on a sprint-by-sprint basis.

Most agile software testing teams do not create automated regression tests. Sure, they think they are, and the scripts they are writing are better than nothing, but they are far from the goal of catching regression defects and ensuring the code is fully exercised.

Why?

Well, most developers hate creating automated tests. Just ask one. Developers want to write software, not tests. Most don't even make complete unit tests, and take every shortcut they can find. The less time spent creating tests, the happier the developer and greater the illusion of productivity (illusion because the bugs are still there and will need to be fixed at a later date, for a higher cost).

If you have a really good, well trained, and diligent software engineer, the best you can hope for are integration tests. Unfortunately, this leaves severe gaps in coverage when compared to how your customers actually use the product.

2. For some teams, automation tests are not used as part of a build acceptance procedure.

We've all been there. You walk in on the Friday before a promised Monday release, and start the build. After 20 minutes, the build fails. As you go looking for the culprit, you find the entire team is doing the same thing.

When your project first started, you didn't need build verification tests prior to code check in. As time went on, build scripts were added on an as needed basis, typically by whoever needed it at the time. Many times no automated build process has been created at all, and the idea of passing 100% automation tests before a build is considered successful has never crossed anyone's mind . . . until you're paying the price for not having one.

Then you look at your current automation tests, and most require results to be analyzed before you can say whether something is actually a failure. The concept of automatically created defects, rejected check-ins and build verifications sound great until someone has to stop writing code on your product, and write infrastructure to support your methodology.

3. It is very common for automation tests to get abandoned without planned, consistent maintenance.

Developers make code changes, and test scripts break. Fixing broken tests is not a priority in a high velocity development environment.

And what about superseded and out-dated scripts? Most automation tests are brittle and break with only minor changes to the code base.

Remember the Agile Manifesto Principle: "Welcome changing requirements, even late in development." Doesn't that mean the test scripts you wrote last week might not apply this week? Now what?

Usually, exceptions are made for non-critical failed or superseded scripts. After enough scripts are broken, development teams lose faith in automation, and ignore the results completely.

Simply put, abandoned automation libraries are wasted money.

4. Automated regression test results are not criteria of "shippable code."

Have you ever seen an application decrease the number of requirements? Let's face it, requirements to be regression tested never shrink. Ever.

From the perspective of marketing and sales, development is really a black box. It is rare for a product owner to even know about automation tests, and even if the product owner does, how often do they know what the results are? Or what the results really mean?

When someone makes a huge deal, suddenly automation can become important. For some scenarios, like integration with SAP or Oracle, automation test scripts and results are required.

When it comes down to it, failed automation tests are a predictor of increased support calls. The customer never says, "You need to make sure your automated regression tests are passed before you ship the code." Does that really mean the code is shippable if automated regression tests don't exist or aren't passed? Probably not.

On-Demand Testing for an Agile Development Environment

Many of our clients follow agile development practices and have integrated on-demand testing (ODT) tightly into their release cycles. This provides the agile software testing to match their agile software development and has proven very effective in finding defects much earlier.

* ODT identifies the defects so your customers don't have to
* ODT is available in any time zone and is there when you need it,
* TESTCo's On Demand Testers love this stuff

Will On-Demand Software Testing Work for You?

Probably. It depends.

It will if you:

- Have a web application in a stage that is ready to be tested
- Have an application that is accessible via the internet or installable on a personal computer or server
- Have an agile software team that wants their work tested
- Will spend an hour working with us on our first day and 15 minutes per day thereafter
- Give us real feedback on our work and be honest with us about where we make mistakes
- Really want it to work

Those last two items are actually the most important. Agile software testing is NOT a black box that you shove your program into at the last minute. It is a quick and valuable process that you can use when you need it the most. But, it does require that you do your part – the most important of which are communication, honesty and desire. Without those, Agile software testing won't work for you – or us.

Why TESTCo?

I started TESTCo in 2002 to offer small and medium sized software companies the opportunity to take advantage of all of the benefits of outsourced software testing without any of the risks and hassles. We've done very well doing just that.

As you know, though, the economic times have changed. Budgets are smaller, and the need to build software to increase revenues and reduce costs is becoming more important every day. Everyone is being expected to do more with less.

I'm passionate about the quality and value of our Agile Software Testing Services and wanted to make it more available and more affordable for more clients. Over the years we've delivered software testing solutions to a large number of leading software companies.

So, we took everything we've learned about Software Testing and Quality Assurance and made it available to a wider customer base at a very affordable

price. This new offering strips out the extra engineering processes designed for big corporations and makes it easy to start tonight- on a tight budget and timeline.

Now, instead of taking two weeks to create a solid team of testers, it takes only one day. This allows you to slice what used to be a two-month testing process down to just a single day.

It's the perfect testing package for high-velocity businesses.

We actually started offering our Agile Software Testing Services to a select group of clients in early 2008. These clients were very generous with their feedback, and we've made a number of improvements over the past year.

Our On-Demand Software Testing Services are now ready for every business!

More importantly, the services cost less than traditional testing, require fewer people and provide real value the very first day.

What Do You Do Next?

I hope you've found this report valuable.

If you need help today, then by all means, please call us at 888-254-9709. If I don't answer, please leave a voicemail letting me know that you need help urgently. I'll call you back within one business day. You should also send an email to jeff@testco.com so I'll have 2 opportunities to see that I need to get in touch with you quickly.

If you don't need help today but think you might need help in the near future, please send an email to jeff@testco.com and we'll arrange a time to talk on the phone and answer any questions you have.

Here's what one of our On-Demand Software Testing clients said recently –

"*TESTCo was crucial to meeting our testing goals and deadlines. Even though I was too busy to provide proper direction to the testing team, they were able to get started on their own, create test plans based on what I wanted and deliver*

results from day one. As the testing needs of the project grew, TESTCo was easily able to grow with us. They continued to build requirements themselves and they were able to manage the testing process in a way that worked for us, not against us. Bringing in TESTCo is a huge win for any software engineering project." - Apaar

You may find that you like the idea but are still uncertain as to how to proceed. That's not unexpected. Unfortunately, I can't give every detail of every step in the process – it's just too much information. If you find that you're still a bit uncertain, please feel free to contact me at via our website at www.testco.com. We can schedule a quick meeting to answer your questions and discuss the process in more detail.

I wish you the best of luck and success with your next project.

Jeff Hotz, President/Founder, TESTCo

About TESTCo
TESTCo delivers Software Testing Services to carefully selected customers around the world. The Austin, Texas-based company's customers eliminate software testing problems, slash support costs by up to 50% and achieve their release date commitments…all without the risk of hunting down the best outsourced vendor, the pain and frustration of learning how to properly manage them and the unpredictable results so common with most outsourced vendors.