THE ABCs OF AUTOMATED TESTING IN 2018

A 3Pillar Global ebook, authored by Senior Automation Engineering Staff

3PILLAR GLOBAL

Almost every project manager or team lead has the same question at some point in the course of their project - that remains the case when looking ahead to 2018:

What should we be doing in terms of automated testing?

It seems like everyone is getting in on the automated testing game, but what can it do to make your application better or your team more efficient?

What Do We Mean By "Automated Testing?"

While different people have different definitions of what falls under the umbrella of “automation,” we are going to exclusively focus on automated acceptance tests, also sometimes called automated functional tests. As with their manual counterparts, these tests directly validate the acceptance criteria or requirement for the given story.

The acceptance criteria, including a rough outline of how QA plans to test the story, should have been agreed upon by everyone involved prior to beginning work on the story. Therefore, validating that the application's behavior conforms to that criteria means that everyone can agree on the correct application behavior and on whether the feature has been implemented properly if the tests pass.

The goal of automated tests is essentially the same as the goal of manual tests: to ensure that the features being tested are working as expected. Because the goals of both testing suites are the same, they should generally follow the same principles and rough outline of steps to perform that testing. Basically, your automated acceptance tests should be an automated version of the manual test your QA would perform to validate that the feature is complete and working correctly.

Some would argue that unit tests should be considered part of automation. While unit testing is a useful and worthwhile part of the development and QA process, it is examining a fundamentally different aspect of an application. Usually written by developers, unit tests require intimate knowledge of the application's inner workings and often do not have a direct relationship with an individual aspect of the user interface. As such, unit tests should be considered part of good development practices rather than automated testing.

Advantages of Automated Testing in 2018

There are numerous advantages to implementing automated testing for your project, but here are the ones that are the most significant and are likely to be the most impactful.

1 REPEATABILITY

2 INCREASED QUALITY

3 INCREASED EFFICIENCY

Let's take a closer look at how automated testing brings these advantages to life.

1 REPEATABILITY

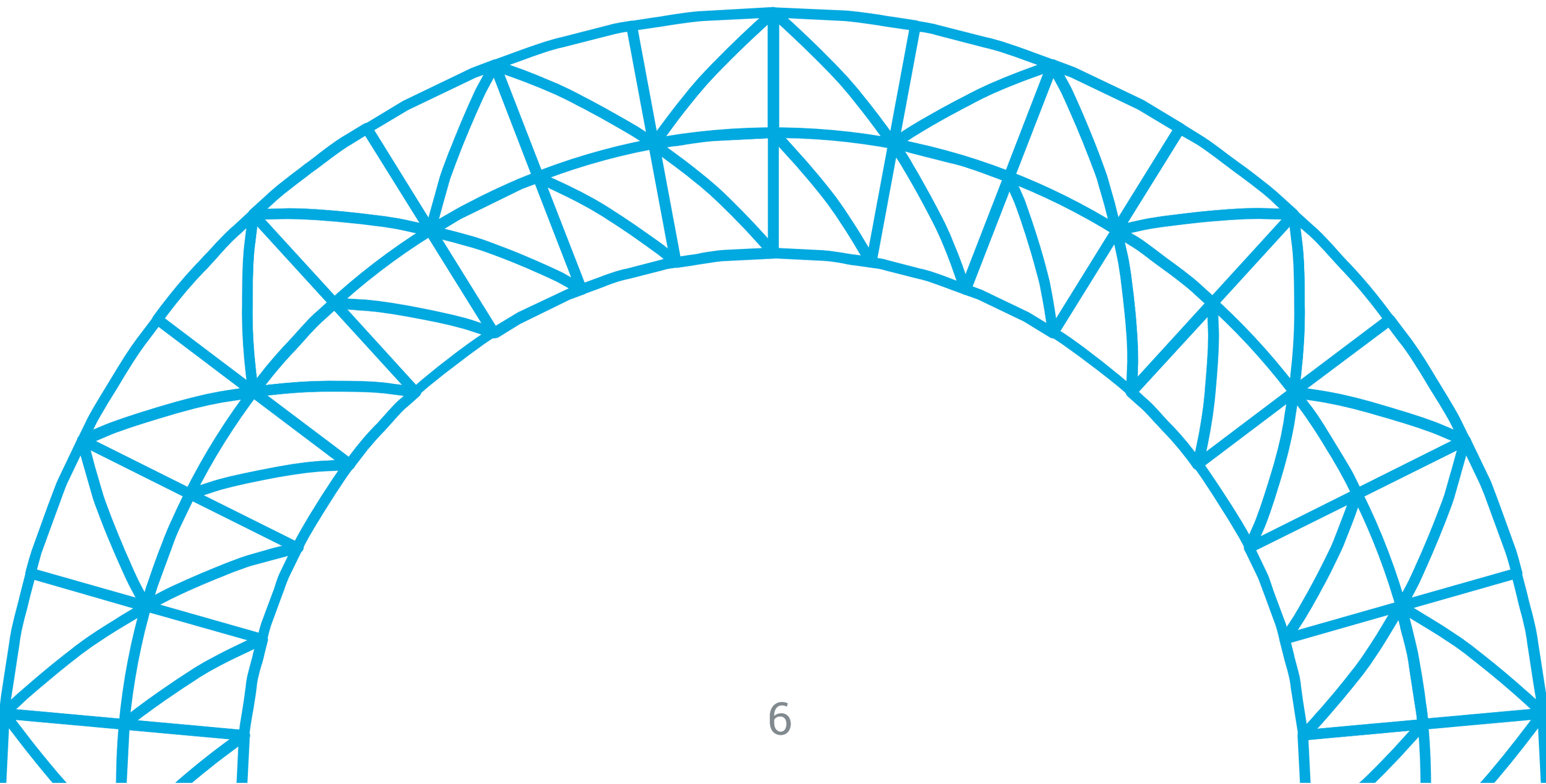
Because test steps are executed exactly the same way on every test run, tests can be executed over and over again quickly and reliably. This is especially apparent when dealing with features that require rote, repetitive behavior to verify, such as making sure a table of data is sorted correctly when sorted by each column. By removing the human factor, test execution and results are more consistent over time.

2 INCREASED QUALITY

While we do everything we can to avoid them, people make mistakes; it’s human nature. Because automated tests are, well, automated, they can be executed over and over again without worrying about mistakes, fatigue, or other human factors that automation doesn’t suffer from. This makes testing more reliable, meaning that fewer defects are likely to slip through the cracks and wind up in the finished product.

3 INCREASED EFFICIENCY

Once you have automation in place, testers performing manual testing work no longer need to perform the tests covered by that automation. This prevents QA from becoming a bottleneck as a project matures, since the usually-lengthy time needed to conduct regression testing for new features is no longer a manual process. This allows the size of the QA team to remain relatively stable even as an application's feature set grows, minimizing project cost and maximizing value to the client.



Disadvantages of Automated Testing

While automating your tests can do amazing things for your project and your team, there are a few challenges to be aware of that automation won't solve.

1 90% COVERAGE, NOT 100%

2 UI LAYOUT TESTING

3 DEALING WITH DATA

4 PLUGIN PROBLEMS

1 90% COVERAGE, NOT 100%

Despite best efforts, it is extremely unlikely that you will ever be able to achieve 100% coverage with automation. There are a number of reasons, ranging from excessively difficult implementation for a particular feature, to technical restraints, to budgetary concerns, but there will almost always be something that simply cannot be automated. Our general rule of thumb is to aim for 90% coverage, following the adage "automate what you can and figure out how to live with what you can't."

2 DEALING WITH DATA

Data consistency is vital in order for your test to be repeatable. Ideally, the tests should start from a known-good and expected state to allow them to access and modify data as need to validate functionality. If your tests are expecting certain data to be present and it isn't, the tests are likely to fail, giving you a false negative. Likewise, you may encounter false negatives if your tests have generated some data and entered it into the application and then another run of the tests attempts to enter the same data.

3 UI LAYOUT TESTING IS CHALLENGING

UI layout testing is particularly challenging to automate. While it would be instantly obvious to a manual tester that a particular button is five pixels lower than the button next to it, automating that sort of validation is very problematic. You could check the properties of the button, or even of its parent element, but there may be an issue somewhere else entirely that is causing it to be pushed down, and finding that issue through automation usually isn't the most efficient method. While strategies like screenshot comparisons can definitely be applied, we find more value in investing in a good design system to minimise UI layout bugs.

4 PLUGIN PROBLEMS

You may have a hard time accessing parts of your application if they are rendered by a plugin. The classic example is a Flash module embedded in a web application. While your automated tests can access the application surrounding the module, most automation tools will be unable to access the Flash module itself, putting the functionality it contains out of reach.

Applying Automation

Consider the following scenarios you may have found yourself in during the project lifecycle. Each of these situations presents a common struggle or pain point for project teams, along with suggestions for how leveraging automation can reduce or eliminate those struggles.

Scenario one: "The New Project"

Scenario two: "The Quality Story"

Scenario three: "The Vicious Cycle"

Scenario one: "The New Project"

Jack is launching a new project and wants to start with his best foot forward. He wants to implement continuous integration/continuous delivery into the project’s processes and wants to make sure that the team gets the most value out of their CI/CD pipeline, as well as ensuring that they can rely upon the quality of the pipeline’s end-product.

Continuous integration/continuous delivery can be a powerful tool in minimizing the time it takes to roll out application changes, a process in which automated testing plays the critical role of ensuring quality. If you were to have a CI/CD pipeline that took in code changes, built them, and deployed them without any testing, you would have no way of knowing whether the application deployed correctly or if your changes were effective. By adding automated testing as a step in that pipeline, you can be confident that your application is in a good state when the pipeline completes.

Beyond simply enabling confidence in your CI/CD process, beginning automation early in the project lifecycle will make a meaningful difference in the impact that automation can make on overall team performance. As previously discussed, automation provides a degree of repeatability, efficiency, and overall product quality that is above and beyond what can be achieved through manual testing alone, and those benefits are exaggerated both the longer automation is in place as well as how early in the process it is begun. Starting your automated testing suite from the ground up along with the project means they are starting fresh together and there is no significant backlog of functionality that needs to be automated in order to get to the ideal level of automated test coverage. The testing team can simply implement automated tests as features are developed and allow the regression suite to grow naturally over time.

Scenario two: "The Quality Story"

Sally is the lead for a project that has been running for several months and is becoming more mature. The team includes several QA engineers who have been creating, maintaining, and executing what has become a fairly comprehensive suite of regression tests.

Unfortunately, as the project has gotten larger and more complex, Sally has noticed that a greater number of issues are being missed and defects are beginning to creep unnoticed into their builds.

The challenges described in this story are the classic symptoms of a project that is crying out for test automation, as the difficulties experienced match the strengths of automation. Removing the human factor improves repeatability, reducing the likelihood that defects will be missed during the course of testing. Automated tests can be executed in far less time than it would take a manual tester to complete the same tasks, improving the efficiency of the entire QA process. Because the tests are more repeatable and more efficient, the QA team can be more selective with how they spend their time and are able to give priority to tasks that cannot or will not be automated. All of this put together leads to a higher quality product without sacrificing extra time or resources.

Scenario three: "The Vicious Cycle"

Joe is the QA lead on a medium-sized QA team that has been performing comprehensive manual testing on their project for nearly a year. They are responsible for completing a large, time-intensive volume of tests on each of their builds and releases and are beginning to find that they have more work to do than time to do it. Joe quickly realizes that in order to continue completing their testing, they will need to do at least one of the following: increase the size of their team, delay releases to give the existing team more time to complete testing, or sacrifice quality by reducing the amount of testing performed by the existing team.

This can be one of the most difficult situations to break out of. Generally, teams that find themselves in this predicament have relied solely upon manual testing to ensure product quality and have now reached a point where that solution is no longer tenable. The challenge here is that none of the solutions mentioned in the scenario are ideal. Increasing the size of the QA team will increase operating costs, and delaying releases and accepting lower quality are a step in the wrong direction. Worse, all those solutions are temporary at best since the problem will only get worse as the project continues to grow and mature.

The increased efficiency of automated tests will allow the QA team to reduce the burden of manual testing and will curb the need for the team to grow or for the project pace to slow. If the problem has progressed far enough that the team doesn't have enough time to automate because they are spending so much time doing manual testing that they wouldn't have to do if they could find the time to automate, and so on, it may be valuable to bring in a second, automation-focused team to assist with getting the automation in place. This allows the original QA team to continue to focus on immediate testing needs at the same time that automation is being put in place, after which the team will be able to spend more of their on the high-level manual testing that cannot or will not be automated.

To Automate, or Not to Automate?

While not everything can be automated and it may not be the right choice in every scenario, adding automation to your QA process can make a meaningful difference in your QA team’s efficiency, your project’s total cost, and your product’s overall quality.

Even though additional effort is required up-front to get automation started, the benefits of automating more than make up for it, and the benefits become more and more pronounced the longer a project lasts. Regardless of which phase of the project lifecycle your project is in, it’s worth asking the question

How can automation help us?

Related Reading

1 HOW TO OVERCOME CHALLENGES FACED IN HYBRID APP TESTING

2 MOBILE AUTOMATION TESTING 101 - TOOLS, FRAMEWORKS & CLOUD-BASED SOLUTIONS

3 DESIGN PATTERNS IN AUTOMATION TESTING

4 ACCESSIBILITY TESTING USING JENKINS AND GOOGLE ACCESSIBILITY DEVELOPER TOOLS

About 3Pillar Global

3Pillar Global builds innovative, revenue-generating software products, enabling businesses to quickly turn ideas into value. 3Pillar balances business-minded thinking with engineering expertise in disruptive technologies, such as mobile, cloud, and big data, to develop products that meet real business needs.

To date, 3Pillar's products have driven over $1 billion in revenue for industry leaders like CARFAX, PBS, and numerous others. Over the course of a decade spent helping clients build industry-leading software products, 3Pillar clients have been acquired for more than $7 billion combined.

For more information on the company, please visit http://www.3PillarGlobal.com.