froglogic

Customer Success Story

Froglogic Squish, the UI automation package

My name is Ben Gilbert and I'm a software developer here at Global Graphics. Most of my time is spent developing our C# application SDK, the gDoc Application Platform. This post gives you some information about one of the tools that the gDoc team uses to automatically test the user interface (UI) of our Windows desktop, Android and iOS products.

Froglogic Squish is a cross-platform UI automation testing tool that has made our lives so much easier when it comes to being 'agile'.

Before we integrated a UI automation framework into our development process, it was very difficult to 'be agile' – I mean in the sense of being able to release at any time, or at least at the end of every sprint.

Our gDoc Application Platform SDK is very UI intensive and without any way of automating UI tests, our QA team would have to manually perform each test at the end of a release. As the product expanded, this 'release phase' was obviously getting longer and longer. It was at a month by the time we decided we needed a better way of doing things because, for a 4 month development cycle, it was unacceptable to spend a quarter of the time manually regression testing.

The solution that we found was Squish. We tried a few UI testing frameworks and Squish was by the far the best fit for us. As a developer in test at the time, I was responsible for implementing a system that would give our team (and management) confidence in the products and confidence that we are able to release at any time. A much more agile philosophy.

Squish features a full integrated development environment (IDE), in which you can create all of your tests programmatically. The IDE has all of the features you would expect: breakpoints, trace points, variable inspection, syntax highlighting, auto-completion etc. As a non-tester, this was perfect for me as I could work in a familiar development environment and rapidly put together automated tests using JavaScript.

The plan was to establish a base framework to wrap all our common user interactions into method calls and have some customised assertions that could pass or fail the test (much in the same way as a unit testing framework). This method would mean that, once the framework was in place, tests in the future could be rapidly created using human readable method calls like:

open_document(<DOCUMENT PATH>);

Do something…

save_document(<SAVE PATH>);

close_application();

Going forward, the framework would have new interactions added in parallel to them being developed in the products. The cross platform nature of Squish means that the base framework can be platform independent as the actual interactions are similar; click this, tap this etc. This helps form a platform independent foundation with platform specific tests as a top layer.

Squish also has the ability to record tests. By this I mean recording the user interactions and then auto generating a JavaScript test that you can play back. This was good for any non-developers who wanted to do some test automation as they could record their test as a user and the JavaScript could be cleaned up later. This almost always had to be done as our automated tests ran on a lot of different systems with different operating system versions and the pure recorded scripts would often fail if run on a machine that did not create them.

So now we develop automated UI tests as part of a user story, in the same way as unit tests. They all run on our build system every time we push a commit and because of this we have eliminated our need for a regression testing phase at the end of a project. QA can now concentrate on exploratory testing instead of mundane repetition of manual regression tests.

Once the UI components are mapped, all testing can be performed via scripting languages.

Valuable Features:

The most important thing with an automated GUI testing tool is the ability to quickly and accurately identify the UI components you want to script up.

Squish, after evaluating quite a few alternatives, was the best at doing this (we have a C#/WPF application).

Another feature, which has been the best for us, is the fact that once UI components are mapped, all testing can be performed via scripting languages.

When I was working as a developer in test, I wrote an entire GUI testing framework in JavaScript which was possible because of Squish's ability to map the UI components accurately.

Squish was also easily integrated into our CI system (TeamCity). It had native support for TeamCity and allowed a Squish build step to be created that can run all, or a selection of automated tests. So now, when we have a successful build, all our UI tests are ran on the products which gives product management and the team greater confidence.

Improvements to My Organization:

My company began investigating automated GUI testing because it was something that we did not do at all and the manual testing overhead meant that we were spending more and more time regression testing our products prior to a release.

Now we are using Squish to automate our user stories as they are implemented, therefore removing the need for a regression phase at the end of a release. This frees up our QA team to perform more exploratory testing instead of repeating regression scripts.

Room for Improvement:

The only criticism I have is the IDE needed a bit of polish in version 5.2 (this may have been fixed in the latest versions). For example, I had a few occasions where Squish would fail to hit breakpoints. An application restart would fix this however.

Use of Solution:

I've used it for approximately 12 months.

Customer Service:

Customer service at froglogic is very good. They answered most support emails within about an hour. They were also happy to provide engineering builds if they had a fix to a problem that was not in a released version yet. They also make themselves very available if you need a call with a representative to explain new features etc.

Previous Solutions:

This was our first UI automation tool but we did evaluate others. We chose Squish because of its scripting ability, instead of just recording a test. Also it was the best tool for UI component identification in our product.

Initial Setup:

Installation is via a self-extracting exe. Getting up and running is very simple, start the IDE, pick your scripting language, point the Application Under Test at your products .exe file and you can start working. For our build agents, installation is the same and the TeamCity integration handled the set-up.

Implementation Team:

We implemented in-house. No problems at all in getting our workflow up and running. Any problems that may be encountered will be diligently handled by Frog Logic support.

ROI:

The biggest ROI is the time that we got back from our QA team because they didn’t have to manually regression test our product every release. They can now concentrate on exploratory testing and doing what a QA engineer should be doing, not just following manual test scripts.

It also means that, because we no longer require a regression phase in a release cycle, we are releasable at any time in the true agile sense. As long as our build system is green, we’re good to go.

For licencing, we purchased an unlimited licence that can be used on as many machines as we like with no restrictions.

Disclosure: I am a real user, and this review is based on my own experience and opinions.

Source:

http://blog.gdoc.com/froglogic-squish-the-ui-automation-package/
https://www.itcentralstation.com/product_reviews/froglogic-squish-review-35886-by-ben-gilbert

Request your free 30 day Squish evaluation

- Learn more about Squish
- Other Squish Resources (videos, tech articles, user communities and more)