



QF-TEST

Professional GUI Testing for Java and Web

QF-TEST Evaluation Report of R.J. Lee Group

Why we chose QF-Test over Squish

by Logan White Stack

July 2009

The QF-Test team asked me how we chose & use QF-Test here at RJ Lee Group. I researched about a dozen GUI testing programs; only QF-Test and Squish looked like they could meet our needs. I downloaded evaluation versions of both, configured both for our system, and wrote a couple tests in both. I explained both to our manual tester, and let her try them. I also explained both to our lead developer. Then, the three of us discussed it, found that both would fill all our technical needs, then each of us individually chose QF-Test over Squish.

Here`s why:

- **I was able to install QF-Test without a single tech-support e-mail because it has better documentation & code quality. (The manual continues to be very useful, it's nicely searchable when I need to reference something)**
- **QF-Test's widget/component database is more organized than Squish's, and allows refactoring widget names with greater ease. This was important to us because our application has many fields, which change name & function as business processes change.**
- **Our team lead appreciated that the scripting languages in QF-Test were JVM languages, unlike Squish.**
- **Our erstwhile manual-tester, now automating-and-exploring-tester, was more comfortable with QF-Test's interface for recording clicks.**
- **The largest deciding factor was that QF-Test may sometimes force-focus the SUT, but nobody liked that Squish sent the scripted events to whatever window the OS currently gave focus. With QF we could multi-task while the script ran in the background.**

We're automating our testing faster than new functionality is added, so testing increases with every release. That's much better than the manual testing document that got longer each release :-)

I used Jython scripts to time test actions like searching, opening records, and application startup. The timing results (with build number & host) go in a database, so we can track performance with vastly greater granularity. We also use looped actions to find resource leaks.

I really like scripting in the JVM. Being able to store the contents of a text field in a variable for a later check is great. So is knowing how many items are in a table before and after a filter is applied. As QF-Test becomes more settled into our development environment, I'm spending more time writing tests in Jython. That's the area where I think QF-Test could use some polish. I love what it can do now - but can your engineers get the scripting integration to do anything else cool? I'm doing things like opening a series of records and finding the average time-to-open, changing each field in a form and checking for exceptions in the log, trying to break the filter by running permutations of search strings, etc.

You guys make a great tool. Thanks,

Logan White Stack
of R.J. Lee Group