The view from

Continuing John Kent's series on test automation

Ghost in the machine: automated testing Part 2: Starting test automation projects

THE ARTICLE IN THE LAST ISSUE of Professional Tester upset a few of you out there in the magical land of test tools, which is most gratifying. The article was not saying that automated testing doesn't work. Rather, the point was that record and playback as a method of building test automation has limits that make it impracticable for anything but very small-scale tests. The limits are hit quite early and occur when you no longer have enough budget/people/time to maintain the recorded tests.

If you are going to invest a great deal of time and money in automating your tests then you are likely to want a return on your investment which will be a large number of tests automated - otherwise it won't have been worth it. Use record and playback and you will have many thousands of lines of automation code for even a moderate sized system's regression test pack and the code will be impossible to maintain without enormous, almost certainly prohibitive, effort. So record and playback as a serious method for building automation has largely been discredited. Don't just take my word for it; there are plenty of articles on the web about this (e.g. Brett Pettichord's web page at www.pettichord.com).

By the way, we are talking about script based tools here. Record and playback does work with mainframe tools like TPNS, QAHiperstation and Verify, but these work in completely different ways to the tools we are concerned with here: WinRunner, Robot, QARun, SilkTest, etc. These tools are script based, where automated tests are run using scripts.

So what is a script? 'Script' is another word for Program. They are the same thing. In my experience programs are called scripts in software tools when the marketing department doesn't want you to know that you have to program them. These test tools have languages that are usually 'VB like' or 'C like' and contain all the standard programming structures such as statements, variables, conditionals and operators. The script-based test automation tools are, in fact, program development environments with compilers, code editing and debugging facilities like any other development language. What makes them test automation tools is that they have special functions that can input data into your system's user interface and get expected results back out.

So, let's bring the strands of the articles together. We have seen that the tools are script-based which means that they are really program-based. You have to write at least some of this script, I mean program code, and most importantly, as mentioned last time you have to maintain the code as well (remember maintenance is the key to test automation). Sounds very much like software development. It is! Software test automation using script-based tools is software engineering.

Software test automation is software development

It is so important that this is understood because thinking that the tools work in the mythical record/playback way is the root cause of most of the test automation tool shelfware. Once you accept that Record/Playback is primarily a sales gimmick for script-based automation tools, you have to accept that you will need to do some programming in order to create automated tests. You may use recording as a starting point for program development (to give you the basic user interface object identifiers), but no more than this.

Thus the tools are really program/playback and you need to adjust your thinking accordingly. For a start, giving an automated test tool to test analysts with no background in programming will, at best, result in slow progress whilst they learn how to program. At worst you will get no progress at all.

We testers know better than most that some people make good programmers and some just don't. It takes a certain mindset to be a good programmer. It is the same with the automated test tools. Some testers, who have a programmer's brain, will thrive on building automation code and some will get nowhere. So you need people who can program - not just testers who have been on the tools course. We testers also know first hand that if the system we are testing is of a very poor build quality, it is usually due to inexperienced programmers. Similarly, if you use inexperienced programmers for your automation, you will get low quality, buggy test automation. Bugs?

Kent

Yes, that's right. Because test automation is software development, you get bugs in it. So now you have bugs in the test automation you are building, as well as the software you are testing. It goes with the territory.

Next, once you have the programming minds, they need to build a sound automation architecture. I'm sure you have tested some systems that were badly designed. You can throw good programming minds at a poorly designed system and they do help, but unless you re-build, you still have a poorly designed system. It is the same with test automation. You need the right design/approach/architecture/framework for the automation code and so you need a programming brain that is capable of designing a good software architecture.

Estimating a test automation project

So that's it is it? Just get the right people, right design and you're off? Well, perhaps you need to ask some questions before you kick off the project. Automation should be considered a project in its own right - and a development project at that. The questions ought to be along the lines of: How much effort do we need to put in and what can get out of it? Perhaps we should plan the project as well.

The first question, "How much effort do we need to put into automation development" is seldom, if ever, asked in a sensible way. What tends to happen is that a decision is made to automate the tests. An automation person is employed to come in for six months to a year and do it. Note: no measure of what it is.

Estimating the effort needed to automate regression testing for a system is not straightforward. Beside the fact that test automation is software development and estimating the size of a development project is as error prone as we all know it is, there is also the problem that no single measure will give you the complete answer.

One of the simplest measures of the size of your task is the number of windows or screens in your system under test. We'll be seeing in later articles that automation is generally built to be 'data-driven'. With record and playback the test data is essentially hard coded in the script (program). In the data driven method the data is in separate files and the automation code is repeatedly 'driven' by this data. The many lines of test data may represent many tests and will drive the same automation code many times. Thus, you can add more tests, i.e. test data, without having to write any more automation code.

So, if you have a system with just a few windows or screens as in, for example, some trading systems, you will have a relatively small amount of automation code driven by a lot of data. In this type of system, your programming (scripting) task will be relatively small and you will be able to concentrate your time on creating lots of tests in the form of test data. If, however, you have lots of windows as in, for example, an insurance system, the amount of programming of the test tool will be much greater. Also note that with large systems consisting of hundreds of windows/screens you will almost certainly not automate tests for all windows so your automated regression test pack will be a sub-set of the whole.

Another important measure is the rate of change of the system's user interface (i.e. windows and screens) over each release of the system. As I said in the last article, test automation is about testing the next release of your system - that is a release that has been changed. Changes to your system's user interface mean that you need to change your test automation in order to work with the new release. You should consider maintenance of the test automation from the start of the project. With a brand new system the rate of change will be high and you need to be able to keep your automation in line with each new build. If there are no changes to the user interface or its behaviour, then you won't need to change your automation in order for it to run OK. This is a nice situation to be in for an automation specialist but it is very rare. Usually maintenance of an application means changes to the user interface. I can think of only one project where the user interface was not changed at all. It involved a major re-engineering of the back end of a system. The Y2K regression tests may have involved little change to the user interface, which also made for relatively easy test automation.

Other questions that need to be asked: How much of the regression test will we automate? Which tests do we automate? What do we want from it? This is perhaps one of the most important questions to ask yourself. Do we need to extend test coverage or just to automate a proportion of the current manual tests?

Next issue: Test automation architectures

John Kent is Managing Director of CISS Ltd who specialise in test automation.