June 2011

testing experience

The Magazine for Professional Testers

printed in Germany

print version 8,00 €

free digital version

www.testingexperience.com

ISSN 1866-5705

Improving the Test Process

Accepting and embracing change in automated acceptance tests with Jubula

by Alexandra Imrie

Automated GUI testing has a terrible reputation. If you mention it at a table of testers or developers, then you'll almost certainly hear that "everything breaks every time you make a change" and that "tests are impossible to maintain". Not only does it have this awful reputation, but it is also often misrepresented. All too often the perception of GUI testing is to *test the GUI* – is the button there, is the text field red etc.

GUI tests are not superficial tests

The misrepresentation is usually the easiest to clear up. If we think less in terms of testing the GUI and more of testing *through* the GUI, then the importance of GUI testing becomes clear. Our customers and users will only be able to work with the software using the GUI. Yes, it's important for them that interactions at the unit level are correct, but their scope of reference is the interface they have in front of them. The workflows and use cases requested in the specifications must be executable using the interface, and to be sure of this, we have to test them.

I heard an excellent metaphor for GUI testing. If we think of the application as a pond, then the unit level is close to the bottom of the pond. We can get very close to the intricate details, but are so far away from the surface that we can easily lose the big picture. Just testing the existence or properties of components in the GUI itself is the equivalent of skimming a stone across the surface of the pond. An acceptance test through the GUI, on the other hand, can be thought of as taking a large stick and poking all the way down to the depths of the pond. Starting from the user perspective, we go right to the bottom, passing every layer of business logic on the way. If there's a problem, then we can send a diver to take a closer look.

Change is normal

The terrible reputation may take some more work to clear up. The two main quibbles noted above are both related to changes. The first complaint is that changes in the code require changes to the tests. If we look at this objectively, we find this is a completely normal and desired part of the development process. It is natural that a change to a requirement may require a change in the code. In the same way, a change to requirements or code may also require a change to the tests. No test at any level can be written just once and yet remain up-to-date despite changes. If we accept that our tests need to be updated to keep in line with continued development, the only question is how much effort that change requires.

A question of effort

Minimizing the effort required to update systems isn't a unique concern of testing though. If we look at the best practices we know from software development, many of them are in place to ensure that any maintenance work is easy to do. Just some of these best practices include:

- naming items to make them easily understandable
- combining instructions to make logical structures
- parametrizing information within structures to make them more generic
- referencing structures instead of rewriting or copying them

If we stick to these best practices, then when it comes to changing something we know that:

- we can search for it and understand it based on its name
- we can add, change and remove instructions or parameters from logical structures and have these changes propagated to all places where the structure has been referenced
- existing logical structures may provide the basis for newly required logical structures

These best practices for software development are just as relevant and usable for automated testing. If tests are easy to maintain, then they can continue to run despite changes to the software. Continuously running tests deliver constant information which feeds back into the process to inform us of any questions, issues or errors from that all-important customer perspective (Figure: quality_curve.jpg).

Where the best practices get forgotten

One of the reasons for the bad reputation of automated GUI testing is almost certainly the predominance of the capture-replay method. While the aim to let the whole team participate in test automation is laudable, a recorded script is by no means an auto-

mated test. It is full of repeated actions, redundancies, mistakes, hard-coded data and implicit assumptions. Although the actions are recorded, the intentions and intelligence aren't. In short, none of the best practices we strive to uphold for our software are in place. Although the script may run once or even a few times, it will soon break. As mentioned above, changes are normal – it is the effort associated with them that is critical. Looking at a recorded script, it is painfully obvious that adapting it to make it maintainable is not very different from rewriting the script from scratch.

Writing tests in program code, however, requires a skilled software developer. The focus on the user perspective is quickly lost and managing the test code can often become a project in itself.

The secret behind successful long-term GUI tests is to preserve the focus on team collaboration to ensure that the user perspective is constantly represented. At the same time, we have to get rid of the effort associated with writing and maintaining program code whilst conforming to the best practices we know from writing software projects.

Introducing Jubula

Jubula is a newly created open source project at the Eclipse Foundation for automated acceptance tests that can be written by all team members according to best practices, without any coding effort. It was created from the core parts of the commercial tool GUIdancer and forms the basis for the continued development of GUIdancer.

Instead of recording or coding, Jubula tests are written using drag and drop from a library of modules. Each module is an executable action such as selecting, clicking, checking, dragging&dropping or entering text. Modules can be renamed for readability, combined to form reusable sequences and are generic in terms of data to make them more flexible. The GUI object recognition (also a frequent pain point) uses a heuristic algorithm to locate even changed objects and is also managed centrally to minimize any necessary maintenance work.

This approach ensures the longevity of the test. Recurring sequences are stored centrally so that changes are propagated throughout the test, and the test structure itself is easy to read because of the natural language naming. The removal of programming activities means that all team members can collaborate on the test automation and also reduces the effort associated with maintaining tests.

A test automated with Jubula is a repeatable verification that a specific workflow functions correctly through the medium that the customer will also use. The intelligence of a manual tester is incorporated via various check actions, dynamic synchronization options and structures to react to planned or unplanned deviations within the workflow. It can be used as an acceptance test to (help) confirm that a feature is complete, and as a regression test to ensure that already completed features are not adversely affected by new developments.

Embracing the change

Change is inevitable, but this shouldn't be used as an excuse to ignore the perspective of those who will use the software we are developing. Continuous acceptance testing provides valuable information about the state of the software and uncovers questions and issues that are better treated earlier rather than later. If done correctly, it is a safety net that accompanies each new build or version to provide software that lets the customer work as desired.

Links

http://www.eclipse.org/jubula

> biography

Alexandra Imrie
earned a degree and an MA in linguistics from York University before starting work at Bredex GmbH, where she is a trainer and consultant for automated testing and test processes. When she is at the office, she is a part of the development team, representing the customer / user perspective for software. Alex helps to write user stories, brings feedback from the field and enjoys contributing to the emerging software in terms of testing. Alex frequently represents Bredex at conferences, where she talks about agility and testing from project experience. She is also involved in event organization and customer communication. Two of her main interests in the world of software development are how to make software user-friendly and how to bring agility to testing processes.