Functional Testing

Introducing Jubula

Eclipse Indigo has just been released. One of the new additions to this summer's release is the Jubula Functional Testing Tool project.

by Alexandra Imrie

The Jubula lettii, or Maned Owl, is a member of the Strigidae species of owl and is found in Africa. Like its animal counterpart, the Jubula Functional Testing Tool can most usually be observed working at night and is also a hunter. Instead of small rodents, however, Jubula's prey is information – which areas of the software conform to acceptance criteria and whether there are any differences in the behaviour since the previous night's hunt. Poetic license aside, the decision to choose an owl name for the testing tool project was based on the idea that tests can be watching and observing at times and with a frequency that are unsuitable for humans. Any testing activity that must be manually repeated quickly becomes tedious, expensive and error prone, and is often too late to boot.

Jubula: the background

This being said, a great deal of testing is still performed manually. This is especially the case for acceptance testing, where the software is put through its paces from the perspective of an end-user or customer. The aim of doing so is to establish whether acceptance criteria have been met and whether there are any deviations, problems or errors identifiable. The importance of this sort of information is frequently underestimated. The sooner we can start checking our changing software from the user perspective (and the more frequently we do so), the more chance we have of reacting to the information in a timely and cost-effective way. Whether we've misunderstood a requirement, added a clumsy workflow or introduced an error, we need to know as quickly as possible. This is where acceptance test automation comes into play – critical workflows can be performed at frequent intervals to test new features and inform us of any regressions in other areas of the software.

The Jubula project was born at the end of 2010 to deal with the task of acceptance test automation and was created from around 85 % of the code from BREDEX's commercial testing tool, GUIdancer, which won the Eclipse Award for the Best Commercial Developer Tool in spring 2010. Jubula is a part of the Indigo Release Train and has just graduated from incubation.

Approach

Both Jubula and GUIdancer are based on the same premise: that tests for a program are just as important as the program code itself and should employ the same best practices for their design and execution. However, despite the focus on best practices known from software development, programming effort is not involved in test creation. The aim of this is to separate the testing mindset from the development mindset. Tests can be written completely from the black-box (user) perspective by anyone in the team, whether they have the role of "tester" "domain expert" or "developer".

Testing through the GUI

Jubula tests control the Application under Test (AUT) via the GUI so that the tests reflect exactly the same perspective that a user has when working with the software. Far from being the brittle and unstable automation strategy it is often claimed to be, test automation through the GUI can easily keep up with development if best practices such as reuse and readability are adhered to. For this reason, Jubula offers an alternative approach to the traditional capture-replay method. Instead of recording actions in the GUI, tests are created hierarchically from reusable modules. The steps necessary to make tests robust, intelligent and flexible are transparent and easy to implement from the outset. The result is automated tests that can accompany a project throughout its whole lifecycle.

Architecture & Concepts

Jubula is a client-server application. The client component (Integrated Testing Environment, ITE) is the place where tests are written, executed and analyzed. The ITE runs as a stan-

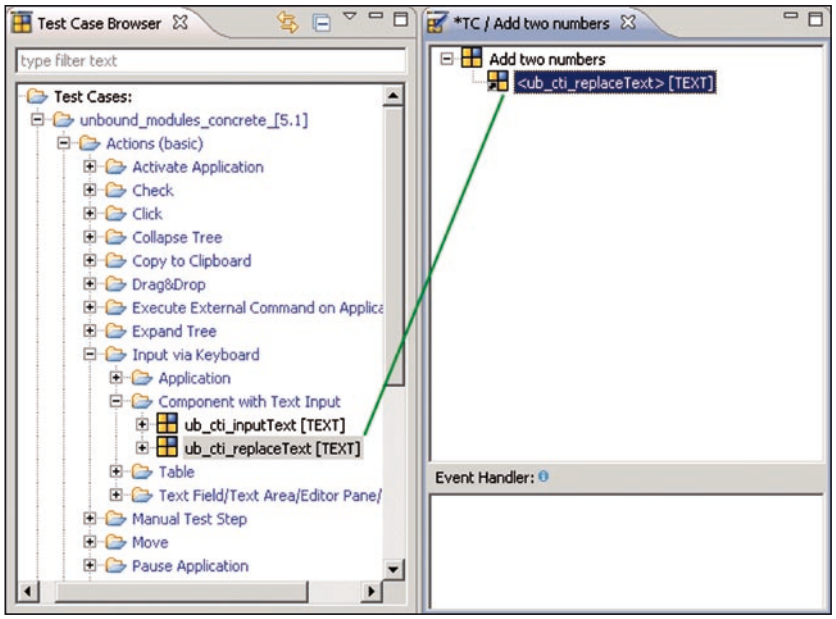

Figure 1: Drag and drop test creation

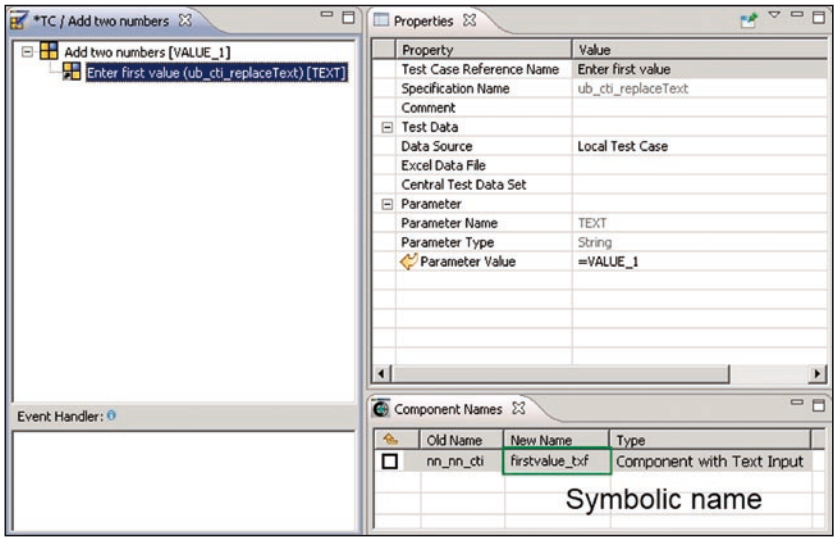

Figure 2: Adding properties and component names for a reused Test Case

dalone application and as an Eclipse plugin. Certain client activities such as starting tests can also be run from the command line (Test Executor) for unattended testing, e.g. with a CI tool of your choice.

The server component (AUT Agent) is a small program that can be installed on any machine within the network. The AUT Agent starts and controls the Application under Test (AUT) and is responsible for the test execution.

Test projects are stored in a multi-user database and can be exported (e.g. to version control) in xml format. The test project contains all the information necessary for a test – from the executable workflows (Test Suites) to individual reusable modules (Test Cases) and all the data they require. Test results are also stored in the database alongside screenshots of any errors that occurred.

Test projects additionally have the capability of being reused. This underpins Jubula's approach to test creation – various generic libraries of actions are included in the distribution from which meaningful modules for various applications (currently Swing, SWT/RCP/GEF and HTML) can be created. Test automation is based on reusability – each Test Case can be reused (referenced) in other Test Cases via drag and drop. The potential to be widely reused can be improved in various ways for each Test Case. In this way, any changes in the AUT can be reflected in minimal central changes to the test. This library-based approach to testing is Jubula's strength. As well as improving productivity due to easy test creation and maintenance, Jubula tests are also readable and can even be created before the AUT is available.

New in Indigo

Jubula 1.0 is a part of the Indigo release train. As the first Eclipse release for the project, the new and noteworthy section could cover pretty much everything available in the tool that has been added over the last 5 years of development as GUIdancer. Instead of reproducing the documentation, it's easier to talk about highlights that should be looked for in the Indigo release. The quick tour of Jubula can be split into three sections: test creation/maintenance, test execution and test analysis.

Test creation

Four steps are necessary to write a new test. First of all, the actions required are selected from the library and added to the test via drag and drop (**Figure 1**).

Once the actions have been added, their names can be made more readable. The third step is to add data – either as specific values, variables or taken from a central test data set. The final step for test creation is to specify a symbolic identifier for the component to be tested. This component name is used later to join the test specification to the actual AUT in a central object map (**Figure 2**).

Once tests start to grow, Jubula offers the option of categorizing modules as well as various search possibilities to find Test Cases. A refactoring function makes structuring tests more comfortable.

Test execution

The prerequisites for test execution are complete data and the object mapping (the joining of symbolic names to actual objects in the AUT). Object mapping consists of "collecting" the information about selected components from the AUT while a special mode is active to highlight the objects with a green border. Component (symbolic) names are joined to their corresponding technical names via drag and drop (**Figure 3**).

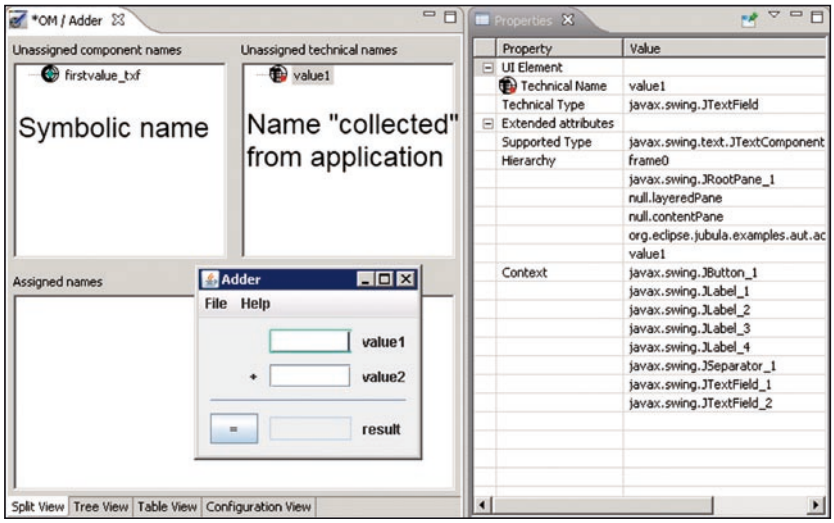

Figure 3: Object mapping

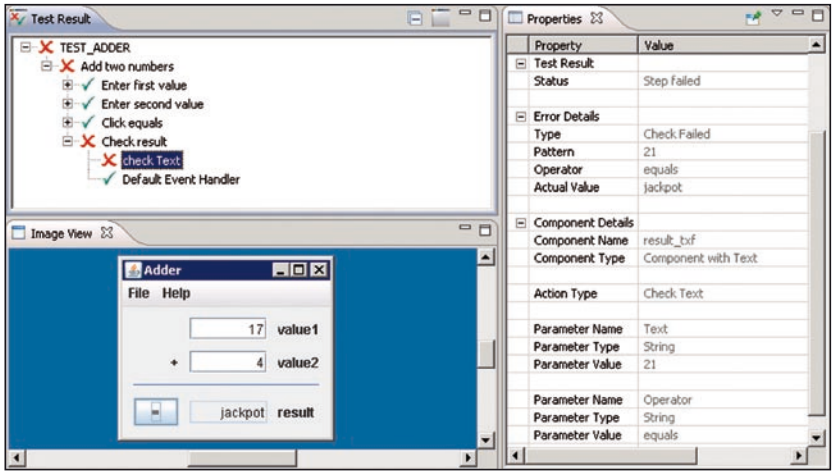

Figure 4: Test Result Analysis

Once these steps are done, tests can be started via the ITE or the test executor. Should a test encounter an error, Jubula offers Event Handlers which allow testers to specify how deviations should be dealt with to maximize test coverage and minimize disruptions. The same test can run on different platforms under different configurations due to the client-server architecture.

Test analysis

Test results from executed tests are stored in the database. They can be viewed directly in the ITE and also reopened at a later date. Any failed steps display a screenshot and details of the error to make analysis more comfortable (**Figure 4**).

If the test needs to be interactively analyzed, Jubula can be configured to automatically pause if an error occurs. The tester can then decide whether to ignore the error for this test run or not.

Getting started with Jubula

Jubula is available from the project's homepage www.eclipse.org/jubula. The software can be downloaded as a complete installable standalone application or via an update site. There is also a new Eclipse Package Eclipse for Testers which contains the Jubula plugins. The Eclipse marketplace provides links to the missing aspects for anyone working with Jubula as a plugin.

Although the first Eclipse Jubula release carries the version number 1.0, the corresponding GUIdancer version is 5.1. In the five years since the product was first released, the further development has been constant, and there are a great deal of options available. Nevertheless, once Jubula is up and running, there are a few places where beginners can get started.

In the main Help menu, there is a selection of cheat sheets which guide testers through their first tests. Sample projects for some examples of more complex workflows are also available (installed as standard in the standalone version). The documentation (user manual and reference manual) for Jubula is included as context-sensitive help. As well as task descriptions, concept explanations and information on some more technical aspects, the documentation contains a section on some best practices for test design. Once the basic steps to automate a test have been mastered, it is worth spending some time to learn about the best practices to make sure that your testing process and activities are a success.

Support and community

The Jubula project pages contain further information about the forum, bugzilla and mailing lists. The Jubula team is eager for feedback, comments, and suggestions, as well as patches and contributions. A guide for contributors is available on the Jubula wiki (http://wiki.eclipse.org/Jubula). Professional support and services are also available from the BREDEX team.

The Indigo Release is just the start of what we hope will be a renewed interest and success in testing with, and at, Eclipse. Anyone interested in following our progress or contributing to the project is invited to join in. Happy testing!

Alexandra 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 organisation 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.