# GUI and DB testing

Validation and Verification
University of Rennes 1

Erwan Bousse (erwan.bousse@irisa.fr)

Last update October 3, 2013

The program under test is a little Java/Swing address book. To store data, it uses a derby database (a file in your home folder). Using Window Tester Pro, you first have to do some GUI testing to ensure that the interface reacts properly. Then using DBUnit you must check that the database backend is correctly updated.

## 1 Background

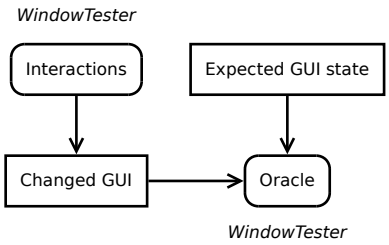
### 1.1 Google WindowTester Pro

WindowTester Pro[1] is a library and an Eclipse plugin that allow developers to easily test GUIs of Java Swing applications. The idea is to give a simple way to manipulate elements of the GUI through a library, and to make the writing of such tests even simpler using a recording/inspecting interface. Generated tests are standard JUnit tests, and can thus easily be replayed.

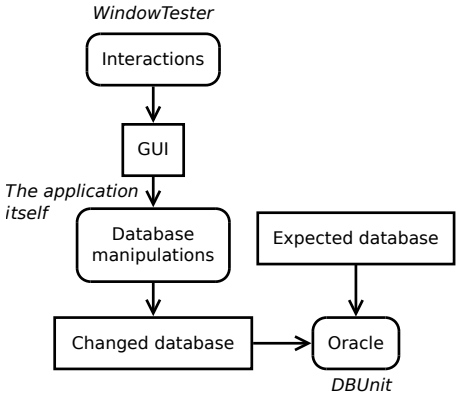
**API Javadocs** `https://developers.google.com/java-dev-tools/wintester/html/reference/javadoc/overview-summary`

**Usage** Before being able to use WindowTester, you must create a run configuration for your Java application. You can do this either by running it once with *"Right click, Run A, Java application"*, or by creating one manually in the "Run Configurations" window. The class to run is AddressFrame. Once you are able to run the application, you can use the WindowTester run button to start the application with the recording/inspecting toolbox. From there, you can start the recording of the actions of the test, then use the inspection tool to make assertions (note that the inspection tool isn't very powerful), and then stop the application and the recording.

[1] `https://developers.google.com/java-dev-tools/wintester/html/index`




(a) GUI testing process




(b) DB testing process

Figure 1: The two testing processes followed in this session

### 1.2 DBUnit

DBUnit[2] is a JUnit extension to test projects that use databases data. This framework solves the problems that can occur when a test corrupts the basis data and then cause a failure of unit tests. DBUnit has the ability to export and import a database to or from an XML file. Since version 2.0, DBUnit can also work with very large data sets when used in "streaming" mode. DBUnit can also help ensure that the database corresponds to values expected for a specific data set.

**Usage** DBUnit allow you to easily create datasets, and also allows you to easily compare them with the actual content of a database. This allows you to write oracles for DB testing. A sample JUnit test class can be found with the code.

## 2 GUI testing

In this part, you will only use WindowTester in order to interact with the GUI for functional testing. Oracles must be based on what the GUI displays after some interactions. Figure 1a sums up the process.

**Preparation** At the beginning of the `pom.xml` file, you must change the value of the `eclipse.home` property so that it points to the folder of the eclipse you are currently using.

**Programming 1** *Using WindowTester, write JUnit tests for the following requirements:*

[2] `http://www.dbunit.org/`

1. *All text fields should be disabled at application startup.*
2. *All text fields should be enabled after clicking on the new button.*
3. *The delete button should be enabled when an item is selected in the list.*
4. *The save button should not be enabled right after clicking on the new button (since there's no data in the fields).*
5. *The delete button should not be enabled when no item is selected in the list.*

# 3 DB testing

In this part, you will use WindowTester in order to add data in the database, and DBUnit to write the oracles. Oracles must be based on what should be in the database, by checking the actual content of the database. Figure 1b sums up the process.

**Programming 2** *With WindowTester and DBUnit, write tests that checks that the creation / modification / deletion of a contact is correctly reflected in the given base. In particular:*

1. *Create two contacts with the same tuple (LASTNAME, FIRSTNAME, Middle-Name, EMAIL) and verify that the contact is not duplicated.*
2. *Check that a new contact with empty required fields cannot be created*

# 4 What to produce

You have to produce an Eclipse Maven project in zip format (Export... → Archive). It must contain:

- The source code of the tested/patched system
- The source code of all your commented tests
- The generated test report (with javadoc)
- The generated test coverage report (with jacoco)