Methods to Improve Quality

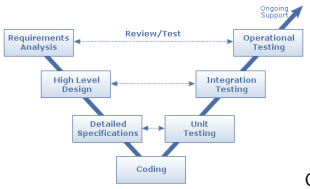
Pieter van den Hombergh,Ferd van Odenhoven

22 maart 2006

1. Validation/(G) User interface testing
   - Validation in development cycle
   - Test scenario sources
   - Test planning; economical point of view.
   - Critical areas
2. GUI Design for automated tests
   - Approaches
   - Gui test frameworks
   - Web interface testing
   - Web test toolkits
   - Canoo webtest
   - Selenium

Testing and the development cycle

- In modern development methodolgy the V-model approach is getting the upper hand.
- The V-model prescribes a balance between specification and verification/validation documents on all level of development.



Of course we expect coding and unit testing to go hand in hand.

Derivation of test cases

- Use cases are the first sources of information for your test scenarios.
- Write the test scenarios during the development of the use cases. This helps you detect missing information and inconsitencies in the use cases. Use this newly found knowledge to improve your use cases and the test scenarios.

Balance in effort

- The amount of automation in your test is matter of proper judgement of effort and gain. These should be balanced.
- Tests that are done 'only once' gain little in full automation. The automation effort is often more then doing the tests by hand.
- In such cases, do these tests by hand, following a test checklist. This checklist could/should be derived from e.g. a use case.

Special attention to critital areas.

One strategy is to automate only very critical areas. These areas need more (often) testing anyhow. Typical examples:

- Authentication and otherwise security related areas.
- Especially test the exception inputs, because they tend to be forgotten in the normal use scenarios.

Aproaches to automated verification/validation

- Robots. Use of record and playback.
- Scripted tests.
- Combination of both.

Robots

- Easy to set up by record and playback.
- (Very) hard to maintain. Especially for the recorders of mouseclicks and keyevents. Any change in GUI design will need a complete re-recording of the test-suite.

Script based gui testing

- The tests are coded in some kind of scripting language.
- The scripts need to 'access' the gui elements.
- This requires the gui elements to be identifiable.
- Consequence: All relevant gui elements should have a (fixed) id (or unique name) over time, so that the id remains stable over versions of the gui.
- The testscript is able to 'grab' the gui element by means of the id handle.
- As a rule of thumb: fix the id's during the analysis phase.

Reference : abbot web page.

A list. Outdated at the moment of writing.

- Abbot
- AberroTest
- Android
- AutoIt
- AutomX
- AutoPilot
- AutoTester for Windows
- AutoTester for OS/2
- Badboy
- CAPBAK
- Certify
- CitraTest
- e-Monitor
- e-Tester
- Eggplant for Mac OS X
- EMOS Framework
- eValid
- Eventcorder suite
- GUIdancer
- IBM Rational Robot
- imbus GUI Test Case Library
- Jacareto
- Jemmy
- QARun
- jfcUnit
- KD Executor
- Marathon
- MITS.GUI (Multiple Interface Testing Suite)
- Perl Win32::GUITest
- Perl X11::GUITest
- Phantom
- Pounder
- PureTest
- Q1
- QC/Replay
- QES/EZ for GUI
- qftestJUI
- QuickTest Professional
- Rational TeamTest
- Ruby Win32::GuiTest
- SAFS (Software Automation Framework Support)
- SilkTest
- Smalltalk Test Mentor
- SQA TeamTest: ERP Extension for SAP
- SQA TestFoundation for PeopleSoft
- Tasker
- Test Now
- TestAgent
- TestArchitect
- TestBench for iSeries
- Tester
- TestGUI
- TestQuest Pro Test Automation System
- TestSmith
- TRecorder
- Unified TestPro (UTP)
- Vermont High Test Plus
- VistaTask Pro
- Visual Test
- WebKing
- WinFeedback
- WinRunner
- X-Unity
- xrc - X Remote Control
- XRunner

Attention points

- Simple HTML pages with forms..
- DHTML and Javascript.
- Differences between browsers. (Minor for display rendering or CSS, a nuisance in DOM handling by the javascript implementations).
- And watch out for AJAX (Asynchrounous Javascript and XML), making web pages both more interactive and more complex to test.

HTML with <form>s.

- Server response could be tested at the server. Response validation is html validation.
- Testable html elements should have names or ids. It allows the test script to validate id and content. Example: <**div id**='topnav_12'>Testable content</**div**> for a navigation element that needs verification in the tests.
- Graphical elements should have alt-texts to simplify element testing. Example: <**img src**='/images/logout.png' **alt**='logout'/>

Example web testing toolkits

- Canoo webtest. See `http://webtest.canoo.com/`
- Selenium webtest with selenium recorder for firefox.
- Watir, similar for Microsoft IE.

have a look at for instance
`http://www.softwareqatest.com/qatweb1.html`

Canoo webtest

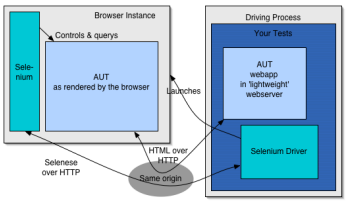
- See handout article from German magazine iX, december 2005 issue.
- Scripting is xml based and built upon Ant.
- Defines project and tasks (as Ant does) and so called steps which define the interaction with and tests of the page under test.
- Test are invoked like antscripts. Could therefor also be used for contineous testing with e.g. CruiseControl.
- Independent of web implementation technology. (Could be jsp+servlets, asp, asp.net, php, cgi/perl, python, ruby etc.)
- For more information and examples see canoo website. `http://webtest.canoo.com/webtest/manual/WebTestHome.html`

Example from canoo webtest testsite

```
<project name="SimpleTest" basedir="." default="main">
 <property name="webtest.home" location="C:/java/webtest" />
 <import file="${webtest.home}/lib/taskdef.xml"/>
 <target name="main">
   <webtest name="myTest">
     <config
       host="www.myserver.com"
       port="8080"
       protocol="http"
       basepath="myApp" />
     <steps>
       <invoke
         description="get Login Page"
         url="login" />
       <verifyTitle
         description="we should see the login title"
         text="Login Page" />
     </steps>
   </webtest>
 </target>
</project>
```

Selenium

- Approach using javascripting
- Script is a simple html table with three columns with special semantics.
- Implementation available for IE an mozilla derivatives (needed because of different approach in DOM handling).
- Recorder for at least firefox is available.
- Uses <iframe>s to display both recorder console, simple test report and page under test.
- Security model of javascript allows manipulation of page elements by javascript in another frame only if both pages originate from the same webserver ⇝ script and page under test must reside on same server.
- `http://selenium.thoughtworks.com/index.html`

Selenium, continued

- The test runner is a simple webpage with a single column table with links to the actual test scripts which are also pages with a table.