SCIENTIFIC PAPERS, UNIVERSITY OF LATVIA, 2012. Vol. 787
COMPUTER SCIENCE AND INFORMATION TECHNOLOGIES 27–49 P.

Self-Testing Approach and Testing Tools

Valdis Vizulis, Edgars Diebelis

Datorikas Institūts DIVI, A. Kalniņa str. 2-7, Rīga, Latvia, *edgars.diebelis@di.lv*

The paper continues to analyse the self-testing approach by comparing features of the self-testing tool developed with those of seven globally acknowledged testing tools. The analysis showed that the features offered by the self-testing tool are equal to those provided by other globally acknowledged testing support tools, and outperform them in instances like testing in the production environment, testing of databases and cooperation in testing with external systems. Furthermore, the self-testing approach makes it possible for users with minimal IT knowledge to perform testing.

Keywords. Testing, Smart technologies, Self-testing, Testing tools.

Introduction

Already since the middle of the 20th century, when the first programs for computers were written, their authors have been stumbling on errors. Finding errors in programs was rather seen as debugging, not testing. Only starting from 1980s, finding errors in a program in order to make sure that the program is of good quality became the main goal of testing. [1]

Since then, software requirements and their complexity accordingly have grown constantly. Along the way, various testing methods, strategies and approaches have been developed. If years ago testing was done mainly manually, in our days, as system volumes and complexity grow, various automated solutions that are able to perform the process as fast as possible and consuming as possibly little resources are sought after.

One of ways of saving both time and resources consumed, improving the quality of the system at the same time, is the system self-testing approach [2]. This approach is one of smart technologies, and it enables the system to verify itself that the software is working correctly. Smart technology is based on the idea of software that is able to "manage itself" by ensuring a control over internal and external factors of the software and reacting to them accordingly. The concept of smart technologies besides a number of significant features also includes external environment testing [3, 4], intelligent version updating [5], integration of the business model in the soft-

ware [6]. The concept of smart technologies is aiming at similar goals as the concept of autonomous systems developed by IBM in 2001 [7, 8, 9].

The key feature of self-testing is the possibility to integrate the testing support option in the system to be tested, in this way ensuring automated testing at any time and in any of the following environments: development, testing and production. To demonstrate the usefulness of the self-testing approach, a self-testing tool that can be compared with globally popular testing tools has been developed. Therefore, the main goal of this paper was, by studying and comparing the concepts, builds and features of various globally recognized testing tools, to evaluate the usefulness of the self-testing approach and tool, directions for further development and opportunities in the area of testing.

As shown in [10, 11], self-testing contains two components:

- Test cases of system's critical functionality to check functions which are substantial in using the system;
- Built-in mechanism (software component) for automated software testing (regression testing) that provides automated storing and playback of tests (executing of test cases and comparing the test results with the standard values).

The defining of critical functionality and preparing tests, as a rule, is a part of requirement analysis and testing process. The self-testing software is partly integrated in the testable system [12, 13], which has several operating modes; one of them is self-testing mode when an automated execution of test cases (process of testing) is available to the user. After testing, the user gets a testing report that includes the total number of tests executed, tests executed successfully, tests failed and a detailed failure description. The options provided by self-testing software are similar to the functionality of testing support tools. Unlike them, the self-testing software is part of the system to be developed. It means that there is no need to install additional testing tools for system testing at the system developers, customers or users.

The paper is composed as follows: Chapter 1 describes the principles used to select the testing tools to be compared with the self-testing approach. Also, the selected testing tools are described in brief in this Chapter. Chapter 2 provides a description of the criteria used to compare the self-testing approach and the testing tools and a comparison of them.

1. Testing Tools

1.1. Selecting the testing tools

Nowadays a wide range of testing tools is available, and they are intended for various testing levels on different systems. When developing one's own testing tool,

it is important to find out what testing tools are being offered by other developers and what are their advantages and disadvantages.

Considering that the self-testing tool employs, in a direct way, the principles of automated testing, various automated testing tools were selected for comparing. In selecting the tools to be compared, the opinion of the Automated Testing Institute (ATI), which is a leading authority in the field of testing, was used. Since May 2009, the ATI has been publishing its magazine Automate Software Testing [14], which is one of the most popular in the field of testing and has its own website too [15]. The website offers articles by experienced IT professionals on the automated testing approach, frameworks and tools; the website has a list of 716 automated testing tools available in the world and brief descriptions of them. Also, a closed forum is active; its users are registered only after approval (currently there are about 8,000 users registered).

The ATI organises an annual conference on automated testing, called Verify/ ATI [16], during which new approaches and tools are demonstrated and training on them is provided. Since 2009, the company has been nominating the leading automated testing tools in various categories awarding them with the ATI Automation Honors [17]. Winners of the award are selected by a committee that is composed of IT professionals, and they study the tools (information on tools is obtained from their official websites, documentation, various articles, blogs, forums etc) and narrow down the list of tools applied for the award to five finalists in each category and sub-category (categories in 2010) [18]:

- Best open source code automated unit testing tool; sub-categories: C++, Java, .NET.
- Best open source code automated functional testing tool; sub-categories: WEB, Java, .NET, Flash/Flex.
- Best open source code automated performance testing tool; sub-categories: WEB, WEB services/SOA.
- Best commercial automated functional testing tool; sub-categories: WEB, Java, .NET, Flash/Flex, WEB services/SOA.
- Best commercial automated performance testing tool; sub-categories: WEB/HTTPS, WEB services/SOA.

For the comparison of the self-testing approach and testing tools in this paper, tools that have won an award in one of the aforementioned nominations were used. In the following sub-chapters of this paper, some of the award winners that are most similar to the self testing approach have been described in brief.

1.2. TestComplete

TestComplete is an automated self-testing tool developed by SmartBear; it provides the testing of Windows and web applications and is one of the leading functional testing tools in the world. This is also proven by the fact that the tool has won the ATI Automation Honors award as the Best Commercial Automated Functional Testing Tool in 2010, and it is used in their projects by world's leading companies like Adobe, Corel, Falafel Software, ARUP Laboratories, QlikTech etc. [19]

Concept

The TestComplete tool uses a keyword-driven testing framework to perform functional tests; in addition, with it it is possible to also develop tests with scripts. Its operation concept is comparatively simple. As shown in Figure 1, the tool, through inter-process communication and various built-in auxiliary tools, records the actions performed in the tested system and after that also executes them.

Figure 1. TestComplete Concept

After each test, the tool creates a detailed report on the test execution, showing the results of every command execution and the screenshots obtained during the playback. In this way, TestComplete makes it possible to overview the errors found in the test.

1.3. FitNesse

FitNesse is an open source code automated acceptance testing tool that can be used to create tests in the Wiki environment through cooperation among testers, developers and customers [20]. Wiki is a webpage content management system that makes it possible to create new or edit existing web pages with a text editor or a simple markup language [21].

In 2010, this tool won the ATI Automation Honors award as the best open source code automated functional testing tool in the NET sub-category.

FitNesse is based on the black box testing principles and Agile manifestos:

- People and interaction over processes and tools;
- Operating software over comprehensive documentation;
- Cooperation with the customer over negotiating the contracts;
- Reacting to changes over following the plan.

The goal of this tool is to make acceptance testing automated and easy to create and read also for people without in-depth IT knowledge. Consequently, customers themselves can develop their own tests as the test creation principles are, to the extent possible, tailored to the business logics. The engagement of the customer in the testing process helps to define the system requirements more precisely and the developers can better understand what the system has to do.

Concept

The testing concept of FitNesse is based on four components:

1. A Wiki page in which the test is created as a Decision Table;
2. A testing system that interprets the Wiki page;
3. A test fixture called by the testing system;
4. The tested system run by the test fixture. [22]

In the test development process, only the Wiki page and the test fixture has to be created over. Everything else is provided by the FitNesse tool and the tested system.

Depending on the test system used, FitNesse provides test tables of various types. To demonstrate the principles of how FitNesse works, the simplest test table, Decision Table, is looked at.

If it is required to develop a test which tests the class that performs the exponentiation of a number, then the following decision table has to be created on the Wiki page:

```
|Exponentiation|
|base|exponent|result?|
|2|5|32|
|4|2|16|
|1.5|2|2.25|
```

The Wiki environment transforms the text into a more illustrative format (Table 1).

Table 1

FitNesse Decision Table

ExponentiationTest		
base	**exponent**	**result?**
2	5	32
4	2	16
1.5	2	2.2

When executing the test, FitNesse delivers the table to the specified test system, which interprets it and calls the following test fixture created by the system developer (in this example, the test fixture is written in C#):

```
public class ExponentiationTest
{
   private double _base;
   private double _exponent;

   public void setBase(double base)
   {
      _base = base;
   }
```

```
    public void setExponent(double exponent)
    {
      _exponent = exponent;
    }

    public double result()
    {
      return TestedSystem.ExponentionClass.Exponent(_base, _exponent);
    }
  }
```

The interpreting is done between titles in the table and the test fixture code. The first row in the decision table contains the name of the test fixture class "ExponentiationTest". The second row specifies the test fixture class methods that are called in the same succession as the table columns are defined. For the methods followed by a question mark also the returned value is read and then compared with the expected value. Other methods set up the input data for the test fixture class. When the test is finished, the returned values are compared with the expected values, and the results are shown in the following format:

Table 2

FitNesse Decision Table after Test

ExponentiationTest		
base	**exponent**	**Result?**
2	5	32
4	2	16
1.5	2	2.2 expected 2.25 actual

Using this concept, system testing is generalised to business logics, and therefore the customers can participate in the process as they only have to additionally master the principles for creating Wiki pages and test tables, not a programming language. Furthermore, a Wiki page can be created also as the acceptance testing documentation since the tables demonstrate the criteria that must be fulfilled for the system developed to comply with the needs of the customer.

1.4. Ranorex

Ranorex is a typical graphic user interface testing tool that can be used by both testers and developers to swiftly and easy create new and manage existing functional tests. This tool has been appraised by the Automated Testing Institute: in 2010, this tool won its award as the best commercial automated functional testing tool in the NET and Flash/Flex sub-categories. In the entire category, it won the 2nd place after the aforementioned test tool TestComplete.

Customers that use this tool are globally known companies like Bosch, General Electrics, FujitsuSiemens, Yahoo, RealVNC etc.

Concept

Like all popular modern graphic user interface testing tools, also Ranorex's concept is based on keyword-driven testing, recording the object, action and identifier.

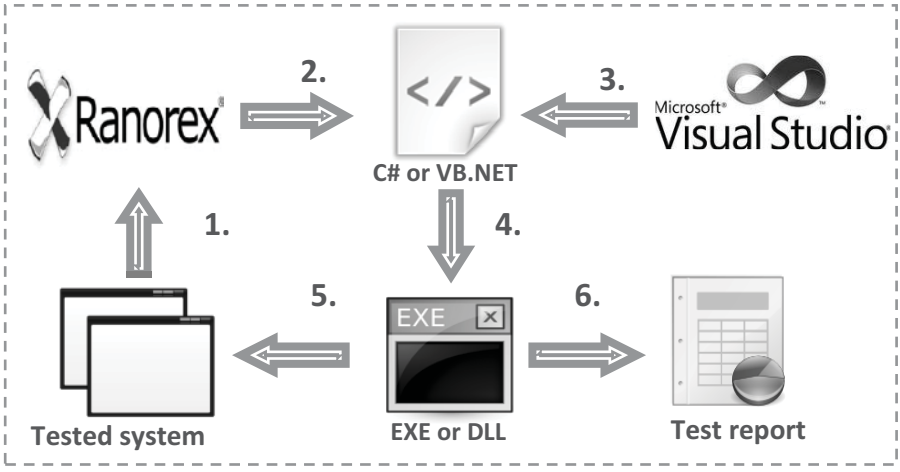

Figure 2. Ranorex Concept

1. Ranorex records the actions performed in the tested system.
2. Ranorex transforms the recorded actions into C# or VB.NET code.
3. To make the testing more convenient also for developers, the created C# or VB.NET code can be edited also in the Visual Studio development environment.
4. An executable file or library is compiled from the code.
5. The compiled testing "program" executes the recorded actions on the tested system.
6. When the test is finished, a test report is generated in which the test status is shown: successful or failed. For each test, detailed information can be viewed (Figure 3).

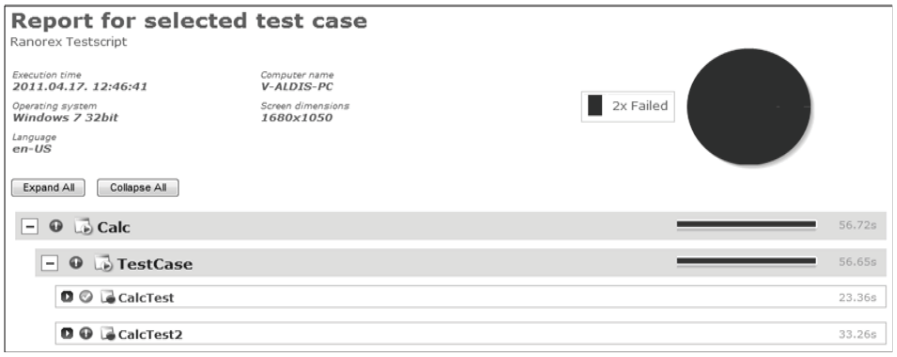

Figure 3. Ranorex Test Case Report

1.5. T-Plan Robot

T-Plan Robot is a flexible and universal graphic user interface testing tool that is built on image-based testing principles. This open source code tool does system testing from the user's perspective, i.e. visual. Since the tool is able to test systems that cannot be tested with tools that are based on the object-oriented approach, in 2010 this tool won the ATI Automation Honors award as the best open source code automated functional testing tool in the Java sub-category. Among the company's customers there are Xerox, Philips, Fujitsu-Siemens, Virgin Mobile and other.

Concept

Unlike many other typical functional testing tools, T-Plan Robot uses neither data- nor keyword-driven testing. Instead, it uses an image-based testing approach.

This approach lets the tool be independent from the technology which the tested system is built or installed on. This tool is able to test any system that is depicted on the operating system's desktop.

T-Plan Robot works with desktop images received from remote desktop technologies or other technologies that create images. For now, the tool only supports the testing of static images and the RFB protocol, which is better known with the name Virtual Network Computing. In future it is planned to add support for the Remote Desktop Protocol and local graphic card driver. [23]

In Figure 4, it can be seen how the testing runs using the client-server principle. The client and the server can cooperate through the network using the TCP/IP Protocol, or locally, using the desktop driver. T-Plan Robot works as a client that sends keyboard, mouse and clipboard events to the server. The tested system is located on the server, which sends to the client changes in the desktop image and the clipboard. T-Plan Robot is installed on the client's system and runs on a Java virtual machine, which makes the tool independent from the platform.

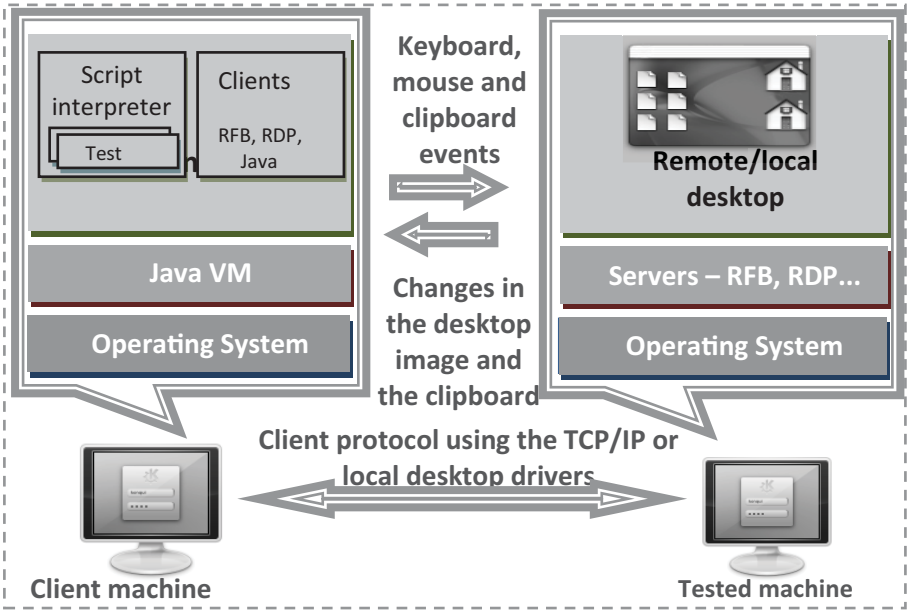

Figure 4. T-Plan Robot Concept [23]

Tests are recorded by connecting to the remote desktop and running the tested system. At this moment T-Plan Robot registers the sent input data and the received changes in the desktop image or the clipboard and creates a test script.

The test is played back by executing the test script that contains the input data to be sent to the tested system. The received changes in the desktop image and the clipboard are compared with those registered when recording the test. At this moment the tool's image comparison methods are used.

The **Client–Server architecture** can be provided in three various ways [23]:

1. One operating system with several desktops: only supported by Linux/Unix as it lets run several VNC servers simultaneously;
2. One computer wits several operating system instances: supported by all operating systems because, using virtualisation technologies (e.g. VirtualBox, VMware etc), the VNC server can be installed on the virtual computer;
3. Two separate computers: supported by all operating systems because when the computers are connected in a network, one runs as a client and the other as a server on which the tested system is installed.

1.6. Rational Functional Tester

The product offered by IBM, Rational Functional Tester (RFT), is an automated object-oriented approach automated functional testing tool that is one of the components in the range of lifecycle tools of the IBM Rational software.

This tool is one of the most popular testing tools, but it has not won any ATI Automation Honors awards. In 2009 and 2010, it was a finalist among the best commercial automated functional and performance testing tools. It shows that nowadays there appear more and more new and efficient automated testing tools to which also RFT is giving up its positions in the market of testing tools.

Concept

IBM RFT was created to ensure automated functional and regress testing for the testers and developers who require premium Java, NET and web applications testing.

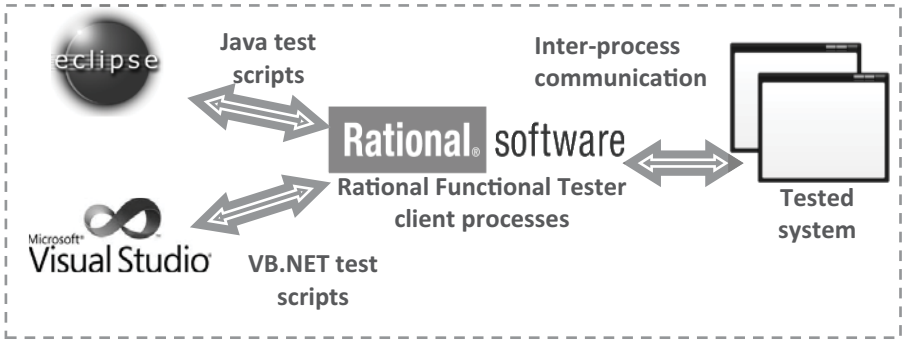

Figure 5. Rational Functional Tester Concept [24]

RFT does not have its own graphic user interface. Instead, RFT uses Eclipse, a development environment that Java users are well familiar with, or Visual Studio that is used by NET developers (Figure 5). In these development environments, RFT adds during installation an additional functionality that makes it possible to record, play back, edit and manage tests. [24]

When the test is recorded, all the actions that take place in the tested system are at once transformed by RFT into Java or VB.NET test scripts. During the recording process, the tester themselves has to create control points for RFT to register the expected system state (e.g. field value, object attribute, system screenshot etc) and later, when the test is played back, to compare that state with the current state.

When playing back the test, RFT executes the test scripts generated during the recording and compares the result of every executed action with the result of the recorded action. If test points are defined in the test script, RFT performs verification in relation to the expected system state defined in the test script. After the execution of each test script, RFT generates a HTML file (log) that demonstrates the test results and shows all the discrepancies for the expected system state.

1.7. HP Unified Functional Testing Software

HP Unified Functional Testing Software (HP UFTS), a tool offered by Hewlett Packard, is a premium quality automated functional and regress testing tool set that consists of two separate tools: HP Functional Testing Software (HP FTS) and HP Service Test Software (HP STS).

HP FTS is better known as HP QuickTest Professional (HP QTP) and formerly also as Mercury QuickTest Professional. HP FTS is based on HP QTP, but it has been supplemented with various extensions. It replaced a formerly popular tool, Mercury WinRunner, which was bought over by HP in 2006. Since most of the tool's functionalities overlapped (or were taken over to) with HP FTS, in 2008 it was decided to terminate the support to the tool and it was recommended to transfer any previously recorded tests to HP FTS. HP STS, in turn, is a tool developed by HP itself, and it ensures automated functional testing of services with the help of activities diagrams. [25]

By merging the tools, HP obtained one of the most popular functional and regress testing tool in the world; in 2009, this tool won the ATI Automation Honors award as the best commercial automated functional testing tool, and in 2010 it was among the four finalists in the same category.

Concept

HP UFTS is a typical keyword- and data-driven testing tool that both makes the creation and editing of tests easier and ensures wide coverage of tests for tested systems.

To perform complete functional testing of a system, it is not sufficient to test the graphic user interface as the system functionality is not limited to solely the visual functionality. Many functionalities are "hidden" under the graphic user interface on the level of components. It can be especially seen in systems that run according to the principle "client-server". These systems are called multi-level systems.

To ensure the testing of such systems, HP UFTS is based on the concept of multi-level testing (Figure 6). HP UFTS distributes multi-level testing in three levels [26]:

- graphic user interface testing;
- services and components testing;
- multi-level testing.

Graphic user interface testing is provided by HP FTS, which divides it into two levels: business processes testing and applications testing. Business processes testing can be done thanks to the test recording and playback functionality and keyword-driven testing that significantly simplify the creation and editing of tests and bring them closer to the business logics. Furthermore, to ensure quality testing of applications, HP FTS, with the integrated script creation and debugging

environment, offers full access to graphic user interface objects and their features. Consequently, HP FTS can be used to test both the process and the GUI level at the same time.

Testing of components and services is done by HP STS, which makes it possible to conveniently create functional tests for the invisible part of the tested system. Tests are created with the help of activities diagrams, and programming is only required for more complex tests; therefore, tests can be created also by users not having knowledge in programming.

Multi-level testing is done by HP UFTS, which combines HP FTS and HP STS in a single solution. HP UFTS can be used to test transactions that unites together the levels of the tested multi-level system. In this way, it is possible, in one test scenario, to test graphic user interfaces as well as services and components.

Test results are generated after the execution of every test in all three testing levels. Both HP FTS and HP STS show test results in two levels. The first level shows general test results and statistics, and the second level offers a detailed description of the results of executing every command and a comprehensive description of every error. HP UFTS gathers the test results in a single report, where all commands are distributed into either HP FTS or HP STS activities.

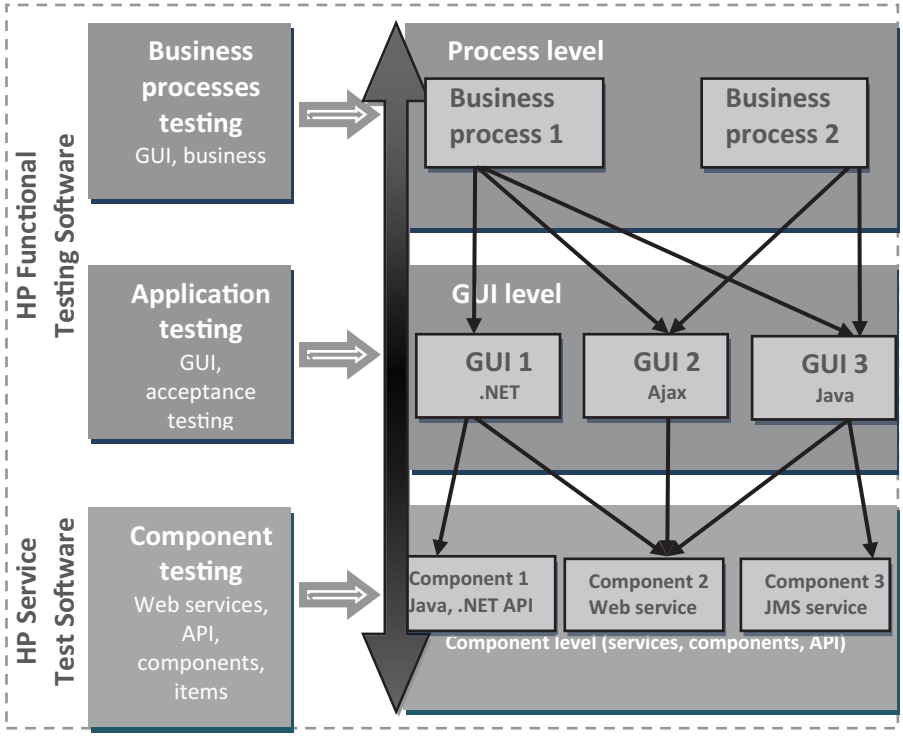

Figure 6. HP UFTS Multi-level Testing Concept [26]

As it can be seen in Figure 6, HP FTS enables the testing of both business processes and applications, making it possible to test the process and the GUI levels respectively in the tested system. Each business process is provided with one or more graphic user interfaces in various setups.

Graphic user interface is the visible part of the tested system that calls various components and services and receives from them the results to be showed to the users. As it can be understood, a majority of system functionalities are located on the system component level, and therefore HP STS offers the functional testing of various services, application interfaces, components and items.

From the aspect of testing concept, HP FTS does not offer a new approach, but in combination with HP STS this tool is able to offer different and diverse functional testing in three levels. This multi-level functional testing makes it possible to perform the testing prior to developing the graphic user interface, in this way allowing a faster development of the system and increasing the quality of components and services. Consequently, the overall quality of the graphic user interface is increased.

1.8. Selenium

Selenium is an open source code web application testing framework developed by OpenQA, and it consists of several testing tools. In the field of web applications testing, this framework has been a stable leader for more than five years, and last two years it has won the ATI Automation Honors award as the best open source code automated functional testing tool. A factor that contributes significantly to the advancement of this tool is that it is used in their testing projects by IT companies like Google, Mozilla, LinkedIn and others.

Concept

The Selenium framework consists of three different tools [27]:

- **Selenium IDE** is a Selenium script development environment that makes it possible to record, play back, edit and debug tests.
- **Selenium Remote Control** is a basic module that can be used to record tests in various programming languages and run them on any browser.
- **Selenium Grid** controls several Selenium Remote Control instances to achieve that tests can be run simultaneously on various platforms.

For test creation, Selenium has developed its own programming language, Selenese, which makes it possible to write tests in different programming languages. To execute tests, a web server is used that works as an agent between the browser and web requests, in this way ensuring that the browser is independent.

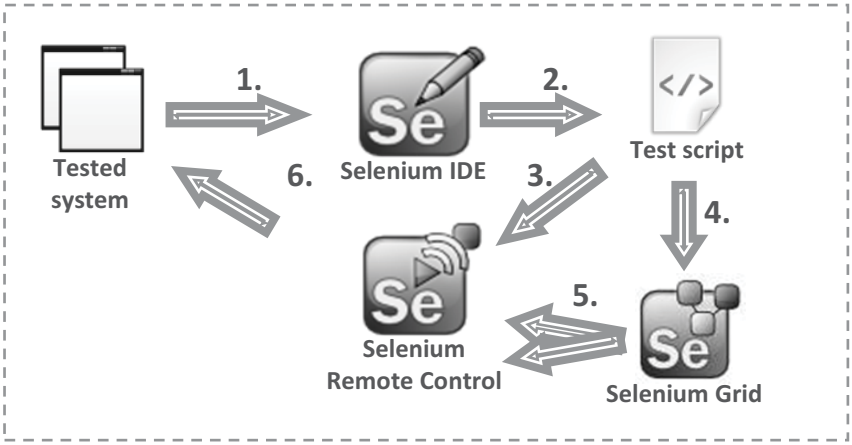

Figure 7. Selenium Concept [27]

1. Selenium IDE, using Mozilla Firefox browser, records the actions performed in the tested system.
2. All the recorded actions are recorded in the Selenese programming language, and if required it is possible to export them to C#, Java, Perl, PHP, Python or Ruby programming languages. After that, using a tool of any of these programming languages, existing script tests can be complemented or new test scripts can be created.
3. The commands recorded in the Selenium Remote Control test script are transformed into web requests.
4. Selenium Grid reads the test script and creates several Selenium Remote Control instances.
5. Selenium Grid simultaneously calls the created Selenium Remote Control instances feeding the read test script as a parameter.
6. Selenium Remote Control forwards to the tested system the web requests that comply with the commands defined in the test script.

It should be added that the concept of the Selenium framework does not require the succession shown in Figure 7. Depending on the knowledge and experience of the tester and the project's needs, the succession can be modified. To create the first tests, an inexperienced tester will certainly use the Selenium IDE development environment. In this case, the testing will be done using the complete cycle (scenarios in Figure 7 – 1, 2, 3 and 6 or 1, 2, 4, 5 and 6). To create more complex tests, an experienced tester will usually skip the first two steps as the creation of tests will be done in a development environment of a programming language supported by Selenium and the execution is provided for by Selenium Remote Control and Selenium Grid (scenarios in Figure 7 – 3, 6 or 4, 5, 6).

2. Comparison of the Self-Testing Tool with other Testing Tools

The aim of this paper is to evaluate what features are offered by the self-testing tool compared to other tools and to identify directions for further development. It is difficult to determine from voluminous descriptions of tools what advantages and disadvantages a tool has compared to other tools. Also, it is rather difficult to assess which tool is best suited for a certain testing project. For this reason, it is important to compare the tools using certain criteria, which are analysed hereinafter and on the basis of which the tools will be compared.

2.1. Criteria for comparison

The criteria for comparison were selected on the basis of the possibilities offered by those seven tools looked at herein. The following aspects were taken into account by the author in selecting the criteria for comparison:

1. key features of testing;
2. key features of automated testing tools;
3. features offered by the compared tools.

Table 3 below lists the criteria used to compare testing tools and the question arising from the criteria, and answers to the questions are provided in the tool comparison tables (Table 4 and Table 5). To compare the self-testing tool and other tools looked at in this paper, the criteria described in Table 3 were used.

Table 3

Criteria for Comparing Testing Tools

Comparison criteria	Question
Test method (TM) [28]	What test methods are supported?
Test automation approach (TAA) [29]	What test automation approaches are used?
Test automation framework (TAF) [30]	What test automation frameworks are used?
Testing level	What test levels are supported?
Functional testing	Is functional testing supported?
Non-functional testing	What non-functional testing aspects are supported?
Platform	What operating systems are supported?
Testable technology	What technologies (usually programming languages) are supported?
Test recording and playback	Is test recording and automated reiterated playback provided?

Desktop applications testing	Is desktop applications testing provided?
Web applications testing	Is web applications testing provided?
Services testing	Is services testing provided?
Database testing	Is it possible to test only the system database separately?
Testing in production environment	Is testing in the production environment provided?
System user can create tests	Can system users without in-depth IT knowledge create tests?
Simultaneous running of several tests	Can tests be run simultaneously?
Performing simultaneous actions	Will the test performance not be disturbed if during the test simultaneous actions are performed?
Identifying the tested object	Is it able to tell apart the object to be tested from other objects of the operating system?
Test result analysis	Is a test result analysis offered after the test?
Test editing	Is an editor for the created tests offered?
Screenshots	Are screenshots of the tested system acquired during the recording/playback of the test?
Control points	Are control points offered?
Object validation	If modifications take place in the tested system, is object validation provided?
Object browser	Is a browser/editor for objects of the tested system offered?
Test log	Is a test performance log created?
Test schedule planner	Is it possible to set the time for performing the test, e.g. at night?
Identification of the end of command execution	Is the tool able to determine when the execution of the previous command has ended (and a certain waiting time is not used for this purpose)?
Plug-ins and extensions	Is it possible to create own plug-ins and extensions to expand the tool's functionality?
Developer	What company or person has developed (owns) the testing tool?
Price	How much the tool costs?
Convenience of use (1-5)	On the scale from 1 (very inconvenient) to 5 (very convenient) – how convenient it is to create tests (author's subjective assessment)?
Tool programming language	In what programming languages it is possible to create tests?
Client	What companies use the tool in their testing projects?

2.2. Comparison results

To assess every comparison criteria determined, information obtained from the tool's official website and other trusted websites, specifications and help windows and from practical use of the test tools was used to make sure that they comply with the respective criteria.

To ensure maximum objectivity of the comparison results, the author tried, to the extent possible, avoid using his own subjective judgment and base the comparison on whether the tool offers a feature or not. To this end, the criteria in the comparison provided in Table 4 and table 5 were evaluated using the following three answers:

- Yes – it means that the tool supports the functionality referred to in the criteria;
- Partially – it means that the tool partially supports the functionality referred to in the criteria;
- No – it means that the tool does not support the functionality referred to in the criteria.

The paper contains two comparison tables: Table 4 provides a summary on the comparison criteria that the self-testing tool supports, and Table 5 provides a summary on the comparison criteria that the self-testing tool does not currently support.

Table 4

Comparison of Testing Tools (Features Provided by Self-Testing Tool)

Criteria		Self-testing tool	TestComplete	FitNesse	Ranorex	T-Plan Robot	Rational Functional Tester	Selenium	HP Unified Functional Testing Software
TM	Black-box	Yes	No	No	Yes	Yes	Yes	Yes	Yes
	White-box	Yes	No	No	No	No	No	No	No
	Grey-box	Yes	Yes	Yes	No	No	No	No	No
TAA	Object oriented	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
TAF	Linear	Yes	No	No	No	Yes	No	No	No
Testing level	Unit	Yes	Yes	Yes	No	No	No	No	No
	Integration	Yes	Yes	Yes	Partially	Partially	Partially	Partially	Yes
	Regresion	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Acceptance	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Functional testing		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Platform	Win 2000	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
	Win Server 2003, 2008	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Win XP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Win Vista	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Win 7	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Testable technology	.NET (C#, VB, C++)	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Test recording and playback		Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Desktop applications testing		Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Database testing		Yes	Yes	Yes	No	No	No	No	Yes
Testing in production environment		Yes	No	No	No	No	No	No	No
System user can create tests		Yes	No	No	No	No	No	No	No
Simultaneous running of several tests		Yes	No	Yes	No	No	No	Yes	No
Performing simultaneous actions		Yes	No	Yes	No	No	No	No	Yes
Identifying the tested object		Yes	Yes	Yes	No	No	Yes	Yes	Yes
Test result analysis		Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Control points		Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Identification of the end of command execution		Yes	Yes	Yes	No	No	Yes	Yes	Yes

Table 5

Comparison of Testing Tools (Features not Provided by Self-Testing Tool)

Criteria		Self-testing tool	TestComplete	FitNesse	Ranorex	T-Plan Robot	Rational Functional Tester	Selenium	HP Unified Functional Testing Software
TAA	Image-based	No	No	No	No	Yes	No	No	No
TAI	Data-driven	No	Yes	Yes	Yes	No	Yes	Partially	Yes
	Functional Decomposition	No	No	No	No	No	No	No	No
	Keyword-driven	No	Yes	Yes	Yes	No	Yes	Yes	Yes
	Model-based	No	No	No	No	No	No	No	No
Non-functional testing	Recovery	No	No	No	No	No	No	No	No
	Security	No	No	No	No	No	No	No	No
	Stress	No	Yes	No	No	No	No	No	No
	Load	No	Yes	No	No	No	No	No	No
	Performance	No	No	No	No	No	No	No	No
Platform	Pocket PC	No	Yes	Yes	No	Yes	No	No	No
	Win Mobile	No	Yes	No	No	Yes	No	No	Yes
	Unix	No	No	Yes	No	Yes	Partially	Yes	No
	OS X	No	No	Yes	No	Yes	No	Yes	No
	Other	No	No	No	No	Yes	No	Partially	No
Testable technology	Java	No	Yes	Yes	No	Yes	Yes	Yes	Yes
	Delphi	No	Yes	Yes	Yes	Yes	No	No	Yes
	ASP.NET	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Flash/Flex	No	Yes	No	Yes	Yes	Yes	Yes	Yes
	Silverlight	No	Yes	No	Yes	Yes	No	Yes	Yes
	HTML	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Ruby	No	No	Yes	No	Yes	No	Yes	No
	Python	No	No	Yes	No	Yes	No	No	No
	Perl	No	No	Yes	No	Yes	No	No	No
	Siebel	No	No	No	No	No	Yes	No	No
	SAP	No	No	No	No	No	Yes	No	Yes
	Other	No	No	No	No	Yes	No	Partially	No
Web applications testing		No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Services testing		No	No	No	No	No	No	No	Yes
Test editing		No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Screenshots		No	Yes	No	Yes	Yes	Yes	No	Yes
Object validation		No	Yes	No	Yes	No	Yes	Yes	Yes
Object browser		No	Yes	No	Yes	No	Yes	No	Yes
Test log		No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Test schedule planner		No	Yes	No	No	No	No	No	No
Plug-ins and extensions		No	Yes	Yes	No	Yes	Yes	Yes	Yes

The next table (Table 6) shows the additional criteria for comparing the tools that were not included in the previous tool comparison tables.

Table 6

Comparison of Testing Tools 2

Testing Tool	Developer	Price (EUR)	Con-veni-ence	Client	Tool 's programming language
Self-testing tool	Datorikas institūts DIVI	*	5	*	C#
TestComplete	SmartBear Software	~ 1 400	3	Adobe, Corel, Falafel Software, ARUP Laboratories, QlikTech u.c.	VBScript, Jscript, C++Script, C#Script, DelphiScript
FitNesse	Robert C. Martin	Free	4	*	Java
Ranorex	Ranorex	~ 1 190	3	Bosch, General Electrics, FujitsuSiemens, Yahoo, RealVNC u.c.	C++, Python, C#, VB.NET
T-Plan Robot	T-Plan	Free	3	Xerox, Philips, FujitsuSiemens, Virgin Mobile u.c.	Java
Rational Functional Tester	IBM	2 700 – 11 000	4	*	Java, VB.NET
Selenium	OpenQA	Free	3	Google, Mozilla, LinkedIn u.c.	C#, Java, Perl, PHP, Python, Ruby
HP Unified Functional Testing Software	Hewlett Packard	3 000 – 10 000	4	*	VBScript, C#

* – no/not available

3. Conclusions

To compare the seven testing tools, the author had to analyse not only their builds but also their concepts in order to assess objectively what are the pros and cons of the self-testing tool and what could be the directions for its further

development. From the comparison of the seven tools, the following conclusions can be drawn:

- The self-testing tool, TestComplete and FitNesse, thanks to the possibility to access the internal structure of the tested system, offers the grey-box testing (self-testing tool offers also white-box testing) method, which makes it possible to test the system more detailed. The other testing tools employ the black-box testing method;
- Among the seven tools looked at in this paper, only one uses an image-based approach as the test automatisation approach. All the others use the object-oriented approach. Consequently, it can be concluded that the object-oriented approach is the most popular for the automatisation of tests;
- In comparison to the other tools described in this paper, the self-testing tool provides for a wide range of testing levels, as TestComplete is the only one that, in addition to unit, integration, regress, functional and acceptance testing, offers also stress and load testing, while the others are limited to only three testing levels. It has to be noted that, considering the ability of self-testing to run parallel tests, it would be comparatively simple to implement the stress and load testing support also in the self-testing tool.
- Of the test automation tools dealt with in this paper, FitNesse is the only one that does not provide the test recording and playback functionality. Instead of it, a convenient creation of tests in table format is offered.
- Just a few testing tools offer such features included in the self-testing tool as database testing, simultaneous execution of tests and parallel actions during testing, whereas almost all testing tools offer the identification of the tested object, test result analysis and the adding of control points to the test;
- Only the self-testing tool offers the possibility to run testing in the production environment and the possibility to create tests for users without in-depth IT knowledge;
- The self-testing tools and Ranorex are the only one that do not offer the feature of adding existing or new plug-ins and extensions;
- The self-testing tool is not the only one that requires additional resources prior to creating tests, as also for FitNesse test fixtures must be developed for successful performance of the tests.

A number of the criteria listed in table 5 and currently not supported by the self-testing tool can be implemented through comparatively minor improvements to the tool. For example, to achieve that the self-testing tool supports performance testing, just a new test point that would control the performance of action execution needs to be implemented.

Self-testing is a new and original approach that does not lag behind other tools, and in some areas it is undoubtedly even better.

IEGULDĪJUMS TAVĀ NĀKOTNĒ

This work has been supported by the European Social Fund within the project «Support for Doctoral Studies at University of Latvia».

References

1. Wikiversity. [Online] [Quoted: 24 January, 2011] http://en.wikiversity.org/wiki/Software_testing/history_of_testing.
2. Bičevska, Z., Bičevskis, J.: Smart Technologies in Software Life Cycle. In: Münch, J., Abrahamsson, P. (eds.) Product-Focused Software Process Improvement. 8th International Conference, PROFES 2007, Riga, Latvia, July 2-4, 2007, LNCS, vol. 4589, pp. 262-272. Springer-Verlag, Berlin Heidelberg (2007).
3. Rauhvargers, K., Bicevskis, J.: Environment Testing Enabled Software - a Step Towards Execution Context Awareness. In: Hele-Mai Haav, Ahto Kalja (eds.) Databases and Information Systems, Selected Papers from the 8th International Baltic Conference, IOS Press vol. 187, pp. 169-179 (2009)
4. Rauhvargers, K.: On the Implementation of a Meta-data Driven Self Testing Model. In: Hruška, T., Madeyski, L., Ochodek, M. (eds.) Software Engineering Techniques in Progress, Brno, Czech Republic (2008).
5. Bičevska, Z., Bičevskis, J.: Applying of smart technologies in software development: Automated version updating. In: Scientific Papers University of Latvia, Computer Science and Information Technologies, vol .733, ISSN 1407-2157, pp. 24-37 (2008)
6. Ceriņa-Bērziņa J.,Bičevskis J., Karnītis Ģ.: Information systems development based on visual Domain Specific Language BiLingva. In: Accepted for publication in the 4th IFIP TC2 Central and East European Conference on Software Engineering Techniques (CEE-SET 2009), Krakow, Poland, Oktober 12-14, 2009
7. Ganek, A. G., Corbi, T. A.: The dawning of the autonomic computing era. In: IBM Systems Journal, vol. 42, no. 1, pp. 5-18 (2003)
8. Sterritt, R., Bustard, D.: Towards an autonomic computing environment. In: 14th International Workshop on Database and Expert Systems Applications (DEXA 2003), 2003. Proceedings, pp. 694 - 698 (2003)
9. Lightstone, S.: Foundations of Autonomic Computing Development. In: Proceedings of the Fourth IEEE international Workshop on Engineering of Autonomic and Autonomous Systems, pp. 163-171 (2007)
10. Diebelis, E., Takeris, V., Bičevskis, J.: Self-testing - new approach to software quality assurance. In: Proceedings of the 13th East-European Conference on Advances in Databases and Information Systems (ADBIS 2009), pp. 62-77. Riga, Latvia, September 7-10, 2009.
11. Bičevska, Z., Bičevskis, J.: Applying Self-Testing: Advantages and Limitations. In: Hele-Mai Haav, Ahto Kalja (eds.) Databases and Information Systems, Selected Papers from the 8th International Baltic Conference, IOS Press vol. 187, pp. 192-202 (2009).
12. Diebelis, E., Bičevskis, J.: An Implementation of Self-Testing. In: Proceedings of the 9th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2010), pp. 487-502. Riga, Latvia, July 5-7, 2010.
13. Diebelis, E., Bicevskis, J.: Test Points in Self-Testing. In: Marite Kirikova, Janis Barzdins (eds.)

Databases and Information Systems VI, Selected Papers from the Ninth International Baltic Conference. IOS Press vol. 224, pp. 309-321 (2011).
14. Automated Software Testing Magazine. [Online] [Quoted: 20 January, 2011] http://www.automatedtestinginstitute.com/home/index.php?option=com_conten t&view=article&id=1276&Itemid=122.
15. Automated Testing Institute. [Online] [Quoted: 15 January, 2011] http://www.automatedtestinginstitute.com.
16. Verify/ATI Conference. [Online] [Quoted: 2011. gada 12. maijā.] http://www.verifyati.com.
17. ATI Automation Honors. [Online] [Quoted: 20 January, 2011] http://www.automatedtestinginstitute.com/home/index.php?option=com_conten t&view=article&id=1262&Itemid=131.
18. ATI Schedule & Selection Process. [Online] [Quoted: 16 January, 2011] http://www.automatedtestinginstitute.com/home/index.php?option=com_content&view=article&id=1283&Itemid=156.
19. SmartBear Software. [Online] [Quoted: 15 December, 2010] http://www.automatedqa.com/products/testcomplete/.
20. Testing Geek - FitNesse Introduction. [Online] [Quoted: 3 April, 2011] http://www.testinggeek.com/index.php/testing-tools/test-execution/95-fitnesse-introduction.
21. Wikipedia - Wiki. [Online] [Quoted: 10 April, 2011] http://en.wikipedia.org/wiki/Wiki.
22. Wikipedia. [Online] [Quoted: 10 April, 2011] http://en.wikipedia.org/wiki/FitNesse.
23. T-Plan Robot Enterprise 2.0 Tutorial. [Online] [Quoted: 17 April, 2011] http://www.t-plan.com/robot/docs/tutorials/v2.0/index.html.
24. IBM Help System. [Online] [Quoted: 19 April, 2011] http://publib.boulder.ibm.com/infocenter/rfthelp/v7r0m0/index.jsp?topic=/com.ibm.rational.test.ft.proxysdk.doc/topics/c_pr_architecture.html.
25. HP - Looking for Mercury Interactive? [Online] [Quoted: 9 May, 2011] https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-10^36653_4000_100__&jumpid=reg_R1002_USEN#.
26. HP Unified Functional Testing software. [Online] [Quoted: 6 May, 2011] https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24^1352_4000_100__.
27. Selenium Web Application Testing System. [Online] [Quoted: 30 April, 2011] http://seleniumhq.org/.
28. Pressman Roger, S. Software Engineering - A Practioner's approach 5th edition. New York : The McGraw-Hill Companies, Inc., 2001.
29. T-Plan Robot - Image Based vs Object Oriented Testing. [Online] [Quoted: 3 April, 2011] http://www.t-plan.com/robot/docs/articles/img_based_testing.html
30. Automatic Testing Frameworks. [Online] [Quoted: 19 April, 2011] http://www.automatictestingframeworks.com/?q=test_automation_framework.