A strategy development for test automation of Intasma®

Jonas Daag

2009-02-02
A strategy development for test automation of Intasma®

Examensarbete utfört i datavetenskap
vid Tekniska Högskolan vid
Linköpings universitet

Jonas Daag

Handledare Jesper Stige
Examinator Bengt Lennartsson

Norrköping 2009-02-02
Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtidens ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsmann i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Jonas Daag
Abstract

Ipendo Systems is a software development company that provides a web-based system for managing Intellectual Properties, Intasma™. The testing process of Intasma™ so far has been done manually and not to the desired extent. An evaluation of two different test automation tools was carried out and a number of difficulties arose when a small implementation was done and that contributed to the conclusion of focusing on how to generate more significant test cases.
## Contents

1   Introduction ............................................................................................................................................. 1

1.1  Background ........................................................................................................................................... 1

1.2  Aim ...................................................................................................................................................... 1

1.3  Research Methodology ..................................................................................................................... 1

2   Test Tools ................................................................................................................................................ 2

2.1  Scope ................................................................................................................................................... 2

2.2  Test Execution Tool Types ................................................................................................................. 3

2.3  Selecting the Test Tools .................................................................................................................... 3

2.3.1  Requirements ................................................................................................................................... 3

Summary .................................................................................................................................................... 4

2.4  Test Tool Candidates .......................................................................................................................... 5

2.4.1  HP QuickTest Professional ........................................................................................................... 5

2.4.2  WatiN ............................................................................................................................................. 5

3   Pilot Implementation ................................................................................................................................ 5

3.1  Test Case ............................................................................................................................................... 5

3.2  HP QuickTest Professional ................................................................................................................ 6

3.3  WatiN .................................................................................................................................................. 7

3.3.1  Problems ........................................................................................................................................ 12
4 Summary .......................................................................................................................... 12

5 Use Case Driven Test Generation ...................................................................................... 14
  5.1 Use Case .......................................................................................................................... 14
  5.2 Activity Diagram ............................................................................................................. 15

6 Conclusion .......................................................................................................................... 17

7 Future Work ........................................................................................................................ 18
  7.1 Intasma™ API and code separation .................................................................................. 18
  7.2 Unit testing ...................................................................................................................... 18
  7.3 Performance testing ......................................................................................................... 18

8 References .......................................................................................................................... 19

FIGURES

FIGURE 1: SYSTEM ARCHITECTURE ...................................................................................... 4
FIGURE 2: HP QUICKTEST PROFESSIONAL ........................................................................ 6
FIGURE 3: HP QUICKTEST PROFESSIONAL "EXPERT VIEW" ............................................. 7
FIGURE 4: MICROSOFT VISUAL STUDIO - ADD REFERENCE ............................................... 8
FIGURE 5: WATIN,CORE ...................................................................................................... 8
FIGURE 6: INTERNET EXPLORER DEVELOPER TOOLBAR ................................................ 9
FIGURE 7: MEMBER VARIABLES ........................................................................................ 10
FIGURE 8: LOGIN PAGE ........................................................................................................ 10
FIGURE 9: DOM OF LOGIN-PAGE ....................................................................................... 11
FIGURE 10: LOGIN CODE ..................................................................................................... 11
TABLES

TABLE 1 TEST TOOL OVERVIEW ................................................................. 3

TABLE 2 TEST CASES .................................................................................. 16
1 Introduction

1.1 Background

This thesis project has been carried out at Ipendo Systems, a company that provides various IT services. The company, which was founded in 2007 as a subsidiary of Ipendo Corporation, has been attracting more and larger clients. This fact has forced the company to recruit additional staff as software developers, consultants and architects to meet the new demands. The one area of development that they have not been able to address is software testing. As the services has turned more complex and new functionality is constantly added, the need for a clear-cut way to test the services is considered necessary.

1.2 Aim

The aim of this thesis project is to find and evaluate different test tools for automated test execution and compose a strategy of how to incorporate the tool in the testing process. However, the aim altered after the test tool evaluation because purchasing a test tool would not improve the quality aspect of the software. Instead I turned my focus to test generation, which in turn would be more beneficial for the quality of Intasma™. The aim was redefined as: propose a generic way to produce test cases.

1.3 Research Methodology

The study has been divided into the following parts

- Literature survey
- Test tool evaluation
- Test strategy

The first part of the study is not included in this report. The purpose of the literature survey is to identify present test methodologies and approaches used today. The second part is a practical part where test automation test tools are evaluated. Finally, the last part discusses how test cases can be derived early in the development cycle based on use cases.
2 Test Tools

2.1 Scope

There are mainly four different test levels for software, which are mapped against different phases of the development cycle as shown in the simplified v-model below [5].

<table>
<thead>
<tr>
<th>Business Requirements</th>
<th>Acceptance Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Specification</td>
<td>System Testing</td>
</tr>
<tr>
<td>Design Specification</td>
<td>Integration Testing</td>
</tr>
<tr>
<td>Code</td>
<td>Unit testing</td>
</tr>
</tbody>
</table>

Test execution and comparators tools are appropriate for acceptance- and system -testing hence these are black box testing areas. Integration- and unit -testing are white/grey box testing phases which need manual execution and supervision [3]; therefore these two testing levels are out of scope.
2.2 Test Execution Tool Types

From my point of view there are primarily four different types of test execution tools. Each tool has different benefits and drawbacks as Table 1 proposes [6].

Table 1 Test Tool Overview

<table>
<thead>
<tr>
<th>Tool Type</th>
<th>Linear Scripting</th>
<th>Structured Scripting</th>
<th>Data/Keyword Driven Scripting</th>
<th>Programmable Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Work</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benefit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technical Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Linear- and structured- scripting tools usually have a capture and replay function which records mouse clicks and keystrokes. All events are logged to a script that can be run at a later time. A drawback is that these scripts lack variables that make them hard and almost impossible to maintain. A data driven tool or a programmable framework allows development of reusable and maintainable code at the cost of more initial work [6].

2.3 Selecting the Test Tools

2.3.1 Requirements

Intasma™ is a web based management system that allows complete control of an organization’s intellectual properties, IP. Typical IP are patents, trademarks, designs, domain names and legal files. Companies, organizations and patent bureaus use the system worldwide [7].
Intasma™ uses multi tier architecture with ASP.NET as the top interface layer, a C#.NET application layer and stored procedures with SQL-Server at the bottom [7], Figure 1.

![Intasma™ System Architecture Diagram](image)

Figure 1: System Architecture

Intasma™ is only compatible with Microsoft Windows and Microsoft Internet Explorer [7]. This fact tells me that there will be no need to test for compatibility with other Internet browsers.

To run Intasma™ the minimum requirements are the following [7].

- Intel® Pentium® III or equivalent processor
- 256MB of RAM (512MB recommended)
- Microsoft Internet Explorer 6.0 or 7.0
- Internet Connection

**Summary**

From what I have learned a successful candidate must be able to test web based systems and the .Net framework. Since Intasma™ only compatible with Microsoft Internet Explorer any browser compatibility tests for other browsers is not required.

I concluded that the two possible ways to go is either with a data driven approach for example key words or finding a suitable programmable framework. Linear or structured scripting tools are fast and easy to get started with but they lack flexibility and need a lot of maintenance to work properly [6].
2.4 Test Tool Candidates

One of the most popular and used test tools for software applications is still HP QuickTest Professional (former Mercury) [2]. If some serious testing is to be done one has to consider this high-end test solution. The second candidate, WatiN, is a promising open source-testing framework for .Net applications [8]. The fact that HP QuickTest Professional is a complete functional testing application and WatiN is a programmable framework does not make them comparable application to application. The evaluation is done to determine which approach is preferable.

2.4.1 HP QuickTest Professional

HP QuickTest Professional has several features including; capture and replay, keyword driven and database verification to mention a few. It also supports a large number of development technologies including next-generation.

2.4.2 WatiN

WatiN, Web Application Testing in .Net is an open source framework for testing web applications [9]. It is developed in C# and integrates easily into Microsoft Visual Studio. WatiN does not have any additional features apart from the fully programmable framework but given the time, one can develop data driven-tests and test reports as well.

3 Pilot Implementation

3.1 Test Case

The Intasma™ developers have created several use cases, which describes different scenarios that are carried out by a user. One of the newest features of Intasma™ is called Add Case which I used during the implementation.

The use case had not been updated accordingly. However, the series of actions was intact but the GUI for the use case had not been updated. This fact did not affect the way the test case was derived because only the actions of the user were of importance.

The test case consists of a number of steps or sub-tests. These steps refer to the user’s expected actions and consist basically of mouse clicks.
3.2 HP QuickTest Professional

Working with this testing software was fast and easy. The application was downloaded from Hewlett Packard’s website [10] and the installation was a typical step-by-step wizard taking you through the setup. To get familiar with the application I started by completing a small tutorial that gave me the basic understanding of the workflow and could from there on implement the pilot, Figure 2.

![Figure 2: HP QuickTest Professional](image)

The approach I used was the built in capture-tool for recording the actions I performed. After recording, HP QuickTest Professional lets you go back and change the parameters from constant data to a keyword driven style to pump large amounts of data through the application. Moreover, the tester has ability to set different types of checkpoints to see if the test passes or fails. Another feature that I think is important for testing business processes is to verify the data sent through the application to the data stored in the database. This was easily done by setting up the connection to the database and let HP QuickTest Professional retrieve the data and compare it with the data used during the test. There is also the possibility to create your own scripts; actually the capture-tool creates Visual Basic scripts for every mouse action which at a later point can be edited in the *Expert View* Figure 3.
3.3 WatiN

Working with this framework was quite a different experience from the first test tool. There was a lot of code that had to be developed just to make this pilot work. To start with I downloaded the WatiN library files from [9]. Another useful tool that is needed for successful development is the Internet Explorer Developer Toolbar (IEDT) that is available at [11]. The WatiN library files come with a typical installation wizard that makes it simple to install. To include the libraries in your project you first have to add references to them as shown in Figure 4.
And finally include them in project you are working with Figure 5.

```csharp
using WatiN.Core;
```

Figure 4: Microsoft Visual Studio - Add Reference

Figure 5: WatiN.Core
The Internet Explorer Developer Toolbar installs like an add-on in the browser window. When browsing a web page, IEDT generates a DOM (Document Object Model) for the page so elements can quickly be accessed and referred to.

Figure 6: Internet Explorer Developer Toolbar

My aim was to build a method library that handles all navigation. In addition, I needed some external data and a report system.

The following example will describe the workflow and how I developed tests with WatiN.

Example: Log in

To start with, I created a new class Browser, which holds an instance of the IE object, marked with a red ellipse in Figure 7. The IE object controls and communicates with the browser, Microsoft Internet Explorer.
The next step is to go to the login page shown in Figure 8 and with help from IEDT find the element that you want to interact with. The login page consists of two text fields and button, which means that the login-method consists of three steps.

- Set User Name
- Set Password
- Press Login-Button

The User Name – text field in Figure 8 is marked with a blue rectangle, which indicates that the element is selected with IEDT. The generated DOM is shown in Figure 9.
WatiN offers several different ways to reference an HTML-element. You can either go with an indexed approach or with one of the Find.By-methods. Using the indices has the drawback that any changes in the GUI of the software under test can easily break the code.

Using the Find.By-methods gives you more options. It allows you to search for a specific value, text, ID etc. The code for the login-method is displayed in Figure 10. The second line that is underlined refers to the selected element, which has an id-attribute set to “txtUserName” in Figure 9.
This is the basic workflow for the navigation. There are however need for some small tests or checkpoints for every method. WatiN offers some methods for checking if elements exist or are displayed as expected.

Finally, I extended the test pilot to become data-driven by creating a tab-delimited file with input arguments and fed the test program with them.

3.3.1 Problems

A number of difficulties arose under the development and probably many more will be discovered if one was to implement a full-scale version. In the previous section I discussed different ways of referring to an element and the best way was to use the Find.By-methods. Three distinct problems came up during the implementation. The first problem was that I could not refer to elements that did not have a unique attribute. I solved this problem with a work around by collecting external data. An example is that I had to create a dictionary that maps country codes against country values.

The second problem was that the attribute changed dynamically depending on user input. This problem I could not solve within the time frame of this thesis. A last problem was third-party components, communicating with these components proved difficult but manageable. One component in particular was a dynamic tree-structure that changed the names of the nodes if another node was created or deleted.

As any other developed piece of code this also generates run time errors. That means that error management has to be included as well. Adding more code will of course add more time to the project that is a drawback.

Finally, I implemented a simple error-log, which records any error and stores the data in a text file. For a full scale version a comprehensive and detailed report will be required.

4 Summary

Both these testing software packages could be helpful for increasing the quality of the software under test. However, I feel that incorporating a test tool should only be done if there is an existing testing strategy and that the testing would in some way benefit from it. I believe that having a good test tool but no test cases would turn the test software to shelf-ware because eventually when the test cases are developed the tool might not be the right one. In this case, I am positive that testing manually with significant test cases would be more useful. By simply running the tests with a test tool would probably result in same tests being run over and over without finding any new bugs [1]. Still, there are
many test tools on the market and I am confident that it would not affect the outcome of the evaluation if I had chosen any other tool.
5 Use Case Driven Test Generation

5.1 Use Case

A use case consists of a basic flow and any number of alternative flows [12]. The basic flow is the expected flow while the alternative flows are cases when the user has done other or incorrect choices. Each flow consists in turn of a number of steps. These steps are expected input from the user. For example, a simple login process could look like this.

Basic flow

b1 typing the username
b2 typing the password
b3 pressing the login-button
b4 the login page is displayed

Alternative flow 1 – wrong username and/or password

afl_1 from B2
afl_2 B3
afl_3 wrong username and/or password
afl_4 B1
5.2 Activity Diagram

The activity diagram helps to detect errors in the system’s flow and is a useful complement to use cases [12]. In Figure 11, more functionality is added if the user is not registered, *Alternative flow 2 – not registered*. Activity diagrams are very powerful when it comes to finding missing functionality.

![Activity Diagram](image)

Figure 11: Activity Diagram

Pulling out all test cases from the activity diagram is straight forward and one can create a simplified activity diagram, Figure 12, if the functionality is complex.
From the simplified activity diagram a test case table can be extracted, combining all possible paths through the use case.

Table 2: Test Cases

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>af1</td>
</tr>
<tr>
<td>3</td>
<td>af2</td>
</tr>
<tr>
<td>4</td>
<td>af1,af2</td>
</tr>
<tr>
<td>5</td>
<td>af2,af1</td>
</tr>
</tbody>
</table>
This example is of course very simple but the method to derive test cases is scalable and can be used for much more complex use cases. There are however aspects of testing that need more consideration such as test variables, testability etc.

6 Conclusion

I learned that test automation tools are not the solution for heavy testing of web applications. The dynamic nature of web applications makes it very difficult to implement and maintain the test code. Another issue I discovered was the communication with other techniques and third party components. Blending different techniques such as Flash animations etc with a server side language proves to tough for the test tools to communicate with. Another aspect of automated testing is time; using either a framework or an off the shelf testing product requires time to learn and create the tests [1]. There is no testing product that is 100% compatible with any software that is to be tested and therefore adjusting the testing software to the application to be tested will require both skill and time. My opinion is to test manually because testing is an activity that demands creativity. Testing is about asking unique questions that will give answers that describes the capability of the software under test. Test automation asks the same question repeatedly and in best cases with different arguments or variables. This will result in that not many new errors or bugs will be detected.

I proposed a generic way to derive test cases from use cases. This way is just guidance and does not consider test variables or testability since the scope would be too big for this thesis. I believe that producing test cases this way will catch the most significant test cases and hopefully the biggest errors or bugs can be detected. There are other ways to produce test cases but in this case I chose to utilize something that was already there, that is to say use cases. I decided so because it would not alter the present development method and it is a relatively fast approach.
7 Future Work

7.1 Intasma™ API and code separation

Create an API for Intasma™ and thereby get an overview of what functionality it holds. It will also be easier to create powerful tests and automate them. One major quality improvement in my opinion could be made if the code would be separated from the GUI. This is actually the only way to really test software functionality. Testing an application with GUI limits the possibilities to test with any input. In addition, some errors are related to the GUI that can make finding and correcting bugs really difficult.

7.2 Unit testing

Creating a unit testing strategy for the developers would probably enhance the quality of the code that is produced. The only testing done by the developers today is best-case scenarios, which from what I learnt, is not sufficient.

7.3 Performance testing

As Intasma™ is a web-based application used by numerous clients globally, performance testing should be addressed. For instance, what happens with response times etc. when multiple clients are connected at the same time? I found a few tools that looked promising in my test tool research and these tools could perhaps help in this area. A good starting point could be a client survey of how they experience Intasma™ in terms of performance.
8 References


7. Strige, Jesper. Notes from presentation of Ipendo-systems and Intasma.2007


