An Ontology-based Automated Test Oracle Comparator for Testing Web Applications

Computer Science

Master Thesis

Sheetal Kudari

May 2011

Supervisor: Marie Gustafsson Friberger
Contact information

Author:
Sheetal Kudari
E-mail: sheetalkudari@gmail.com

Supervisor:
Marie Gustafsson Friberger
E-mail: marie.friberger@mah.se
Malmö University, Department of Computer Science.

Examiner:
Annabella Loconsole
E-mail: Annabella.Loconsole@mah.se
Malmö University, Department of Computer Science.
Abstract

Traditional test oracles have two problems. Firstly, several test oracles are needed for a single software program to perform different functions and maintaining a large number of test oracles is tedious and might be prone to errors. Secondly, testers usually test only the important criteria of a web application, since its time consuming to check with all the possible criteria.

Ontologies have been used in a wide variety of domains and they have also been used in software testing. However, they have not been used for test oracle automation. The main idea of this thesis is to define a procedure for how ontology-based test oracle automation can be achieved for testing web applications and minimize the problems of traditional test oracles. The proposed procedure consists of the following steps: first, the expected results are stored in ontology A by running previous working version of the web application; second, the actual results are stored in ontology B by running the web application under test at runtime; and finally, the results of both ontology A and B are compared. This results in an automated test oracle comparator. Evaluation includes how the proposed procedure minimizes the traditional test oracle problems and by identifying the benefits of the defined procedure.

Keywords: Ontology, web applications, testing, test oracle, and test automation.
Acknowledgements

The satisfaction that comes with the completion of this thesis would be incomplete without thanking the people who encouraged me and whose constant guidance helped to complete this project successfully.

First and foremost, I would like to thank my supervisor Marie Gustafsson Friberger for constant guidance and invaluable support and also for time spent in this dissertation. She has been a great source of inspiration. Her enthusiasm and zeal for work is admirable.

Secondly, I would also like to thank my examiner Annabella Loconsole for sharing her ideas and for the guidance.

I would also like to thank all my friends and colleagues for their support and for the useful comments and tips which indeed paid off on completion of this project.

Last but certainly not the least; I would like to thank my husband and my family for their unconditional love and support and also for inspiring and motivating me in scaling greater heights.
# Table of Contents

Abstract ........................................................................................................................................ iii

Acknowledgements ....................................................................................................................... iv

Table of contents .......................................................................................................................... v

List of figures .................................................................................................................................. vii

List of acronyms ........................................................................................................................... viii

Chapter 1 Introduction .................................................................................................................. 1
   1.1 Problem description............................................................................................................... 3
   1.2 Project idea .......................................................................................................................... 3
   1.3 Motivation ............................................................................................................................ 3
   1.4 Research questions .............................................................................................................. 3
   1.5 Expected results ................................................................................................................... 3
   1.6 Outline ................................................................................................................................ 4

Chapter 2 Research methodology ................................................................................................ 5

Chapter 3 Theoretical background and related work ..................................................................... 7
   3.1 Software testing .................................................................................................................... 7
      3.1.1 Black box testing ......................................................................................................... 8
      3.1.2 Automation testing .................................................................................................... 8
      3.1.3 Test oracle ................................................................................................................... 9
   3.2 Ontologies based software testing ....................................................................................... 11
      3.2.1 Ontology-based testing approaches ......................................................................... 11
   3.3 Web application testing ...................................................................................................... 12

Chapter 4 Procedure for the ontology-based oracle comparator for web application testing ....... 13
   4.1 Test oracle problems and procedure for ontology-based test oracle ................................. 13
   4.2 General architecture of the proposed framework ............................................................... 13
4.3 The proposed framework with OntoBuilder, OntoBuilder parser and OWLDiff..16
4.4 Web application form ..................................................................................................19
4.5 Ontology creation for Finnair website using OntoBuilder ..................................21
4.6 Comparison between two ontologies using OntoBuilderParser ..................24

Chapter 5 Prototype for the proposed framework ..............................................................27

Chapter 6 Evaluation of the proposed framework.................................................................31
  6.1 Benefits of the proposed framework ........................................................................32
  6.2 Limitation of the proposed framework .................................................................32

Chapter 7 Discussion and conclusion ..................................................................................33
  7.1 Future work .............................................................................................................34

Chapter 8 References .........................................................................................................35
List of Figures

FIGURE 1 - A test harness with a comparison-based test oracle ..................................................10
FIGURE 2 - The general architecture of the proposed framework ..............................................15
FIGURE 3 - The proposed framework with OntoBuilder and OntoBuilderParser .......................17
FIGURE 4 - Architecture of customized converter (HTML to OWL) ...........................................18
FIGURE 5 - Reservation form of Finnair airline reservation .........................................................19
FIGURE 6 - Reservation procedure for airlines ..............................................................................20
FIGURE 7 - Finnair airline reservation form in OntoBuilder ..........................................................21
FIGURE 8 - Ontology generated by OntoBuilder for Finnair website ...........................................22
FIGURE 9 - A part of the ontology format created by the OntoBuilder .........................................22
FIGURE 10 - Ontology for the Finnair airline reservation in Protégé ............................................23
FIGURE 11- OWLDiff comparator showing the difference between expected and actual results ...........................................25
FIGURE 12 - Schedules form of Finnair airlines ............................................................................27
FIGURE 13 - Hotels form of Finnair airlines ..................................................................................28
FIGURE 14 - Ontology ER and AR demonstrating Finnair web site .............................................29
FIGURE 15 - Ontology ER and AR demonstrating Finnair forms work flow .................................30
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER</td>
<td>Expected Results</td>
</tr>
<tr>
<td>AR</td>
<td>Actual Results</td>
</tr>
<tr>
<td>WAUT</td>
<td>Web Application Under Test</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Software testing is getting more and more important since faults in the software may cause huge loss of money and business. Burnstien [1] has defined software testing as a mechanism to verify a software application to determine whether the application has been built according to the business specifications and whether it meets the technical requirements. The main idea of software testing is to identify any defects or bugs and fix them before the application goes live. Software testing is categorized into manual and automation testing. In manual testing, the execution of test cases will take place with human intervention. The manual testing is very useful during ad-hoc testing, that is random testing of an application, testing some critical functionality and executing the test cases which are rarely performed. Some drawbacks of manual testing are that it is time consuming, very tedious, requires the investment in human resources and prone to human errors. Dustin et al. [2] have defined test automation as a process to automate the testing process by integration of testing tools into the test environment in such a manner that tests execution, logging, and result verifications are done with little human intervention. Test automation is a concept based on simple capture and playback method which records the application functionality or user actions and play them back numerous times, this will eliminates the need for scripting in order to capture a test. Benefits of test automations are that it is fast, reliable, productive, reusable, and repeatable. Some drawbacks are that it is very expensive to automate the test cases, and you cannot automate everything for example: visual references or screen appearance. Pezzé [3] describes test oracle is used to verify the test result to say whether the test is a pass/fail. This is achieved by storing a test evaluation summary (approval criteria) and comparing it with the actual results in the test oracle, automating this process is known as test oracle automation [3].

A web application is an application which can be accessed on a browser window. According to Di Lucca and Fasolino [4], web application characteristics are: they are highly dynamic in nature, based on a client-server architecture, heterogeneous (that is developed with different technologies and programming languages) and there is no full control on user’s interaction. Because of these characteristics, web application testing is considered to be more challenging than traditional software
application testing. Developers often consider testing last when delivering a web application because it is time consuming [6]. In recent years a variety of tools for web application testing have been proposed. Hower [7] has introduced more than 490 tools, both commercial and free testing tools, for web applications are presented and most of them are designed to carry out load and performance test, web functional/regression test¹, HTML validators and link checkers, website security test, mobile web/app testing. Since web applications are dynamic in nature, testing them is a tedious process. Even small changes in the web application requires user to perform regression testing to ensure that the web application is working as expected. It is an advantage if new improved testing methods and techniques are adopted to automate test result verification to enhance the practice of testing web applications. Wu and Offutt [5] have stated that the problem with the test automation process for web application is that it is very expensive and time consuming to automate the test result verification process.

An ontology is formal and explicit specifications of a shared conceptualization of a domain which can encode domain knowledge in a machine processable format [8]. Happel & Seedorf [9] have discussed that the ontologies have been used in a wide variety of domains and are considered to be an important part of many applications. They can be used in agent systems, Service Oriented Architectures (SOA), software engineering, knowledge communication and e-commerce platforms. They are used for domain knowledge communication and provide semantic based access to the internet. Paydar and Kahani [10] have proposed that ontologies can be used in software testing process by performing two steps: a) Developing the ontology which captures the required knowledge to perform the testing process and application domain knowledge. b) To define procedures for using the knowledge embedded in the ontology.

Though automation testing is already available for web applications, there is need to improve the test result verification to identify whether the test is passed or failed; this still requires human intervention. This process can be automated by capturing the test evaluation summary (approval criteria) in machine processable format and comparing it with the actual results, which can be done using ontologies. Paydar and Kahani [10] have stated that ontology-based test oracle automation automates the testing process and eliminates the human involvement to a great extent.

---

¹ Regression test is defined by Burnstien [1] as a process of retesting the software application to ensure that new changes made to the application does not have any impact on previous working conditions of the application.
1.1 Problem description

The purpose of this study is to minimize the traditional test oracle problems and to eliminate the human intervention to a maximum extent during the test result verification process of web applications.

1.2 Project idea

Goal: To define the procedure for how ontology-based test oracle automation can be achieved and used for testing web applications. This procedure evaluates test results automatically for web applications in order to eliminate human intervention to a great extent during the result verification process.

1.3 Motivations

The test result verification process is not fully automated and still requires human involvement. The test result verification process can be automated by using test oracles. Several test oracles are needed for a single software program to perform different functions. Maintaining a large number of test oracle is a tedious job and sometimes it might be prone to errors. The problems related to the necessity of several test oracles can be solved by using ontologies and it will also minimize the human intervention to maximum extent in the verification process of test result during web application testing.

1.4 Research questions

The main research question involves defining the procedure for how ontologies can be used for test oracle automation of web applications.
RQ1: How can an ontology be created for storing the expected and actual results needed for web application testing?
RQ2: How can the actual results and the expected results, which is embedded in the ontology, be compared?

1.5 Expected results

A procedure will be defined to verify the test results of web application automatically by using ontologies which minimizes human intervention to a great extent.
1.6 Outline

This master thesis report consists of seven chapters. In chapter 2 research methodology is described. In chapter 3 the theoretical background and an overview of the research made in the area is presented with related work. Chapter 4 the procedure for the ontology-based test oracle automation for web application testing is described. Chapter 5 describes the prototype of the system. The proposed framework is evaluated in chapter 6 and this thesis is summarized and concluded in chapter 7.
Chapter 2

Research methodology

This section will give a brief overview of the methodology used during the master’s thesis project. Creswell [20] has defined research methodology, is a systematic way to find solution to a problem or to establish something novel. Types of research approaches include qualitative approach, quantitative approach, mixed approach and design science approach.

Through a literature survey, the background on the application areas of software testing, automation testing, web application testing and ontology based automation testing at different level of testing in different areas were studied. The types of sources that have been used for researching the topic include conference papers, observation, journal articles, books, online websites and others.

The nature of this research thesis is defining a novel approach. It follows the design science approach, which seeks to create innovations that serve human purposes; it is a problem solving paradigm and technology-oriented [21]. This approach has been selected since it involves defining procedure that is set of steps used to perform a task.

The research activity in defining the procedure involves

- Build: Demonstrating that the procedure can be constructed
- Evaluate: Identifying the benefits of the defined procedure.

Here the defined procedure will be evaluated to check to what extent it minimizes the human intervention to verify the test results. Since it is a novel approach, comparison with similar type of procedure cannot be done. Hence advantages of using this procedure will be identified.
Below how each research question is addressed is described.

**How can an ontology be created for storing the expected and actual results needed for web application testing?**  
An architecture is proposed for the procedure of ontology-based test oracle comparator for web application testing. The ontology can be created by developing a customized tool for automatic ontology extraction for web applications or by using a combination of already available tools for automatic ontology extraction of web application.

**How can the actual results and the expected results, which is embedded in the ontology, be compared?**  
The comparison between the expected and actual results stored in the ontology can be done by using the OWLDiff. The OWLDiff takes two ontologies as an input and compares it. It checks whether the two ontologies are identical or different.
Chapter 3

Theoretical background and related work

This part of the report describes background information about software testing, black box testing, automation testing, test oracle, ontology-based software testing, ontology based testing approaches and web application testing and includes the related work.

3.1 Software testing

The software development life cycle, that is the steps involved in developing a software product, includes Requirements analysis, Design, Implementation, Testing, Installation, Maintenance [11]. The purpose of software testing is to identify defects and evaluate before the software application goes live. Burnstien [1] has defined that the software testing is usually done at four levels: unit, integration, system and acceptance testing and each of these level may have one or more sub levels. In this thesis the main focus is on system testing. System testing takes place when all the components have been integrated and integration tests are completed. The goal of this system testing is to check whether the system performs according to its business requirements and evaluates functional behavior and quality requirements [1]. There are several types of system tests such as functional testing, performance testing, stress testing, configuration testing and so on [1]. The task to make sure that the software application works as expected and performs intended functionality, is done by performing the functional system testing. So functional system testing is selected and used in this thesis. There are three testing strategies: black box (responsibility based), white box (implementation based), and gray box (hybrid) [1]. Black box testing is very important to perform to make sure that end applications works as expected, without knowing details of implementations. So black box testing strategy is selected and used in this thesis to perform web application testing which is explained below.
3.1.1 Black box testing

Black box testing is a type of software testing where testing of system takes place with no knowledge of its internal implementation, that is testers do not have any knowledge of how the application is built, how the database and server communication is taking place but the tester only knows what function the software has [1]. The test cases designed to perform black box testing focuses only on a set of input data and its desired output. Software testing can be performed based on the different models. Automation testing is described in next section.

3.1.2 Automation testing

According to Dustin et al. [2], automation testing is a process to automate the testing process by integration of testing tools into the test environment in such a manner that tests execution, logging, and result verifications are done with little human intervention.

Dustin et al. [2] discusses the multiple test activities that can be automatized separately in test automation:

- Find the list of test cases to be automated. Only the test cases which are most frequently executed and which are used for regression testing shall be selected for test automation. The test cases containing exceptional scenario or executed fortnightly shall not be included for automation
- Create the test script to automate the test cases
- Execute the test script and store the expected result
- The test script can be executed one or several times in the future and the actual results will be compared with the expected results
- If the actual and expected results are same then the test result will be pass otherwise it will be fail which will be updated automatically in the test cases.

Gómez Pérez et al. [12] have divided the automation testing into two main parts:

- Driving the program: Involves how the testing program activates the program to be tested and
- Result verification: After the tests are executed, there has to be some method to determine whether the tests are passed or failed.

In this thesis, focus is on automating the result verification process to reduce human intervention to a maximum extent. The test result verification is performed in test oracle, which is described below.
3.1.3 Test oracle

Test oracle is a software that applies pass/fail criterion to a program execution [3]. Pezzé and Young [3] have proposed three approaches for test oracle:

- Judging correctness
  It compares the actual output with predicted output of a program. The predicted output is precomputed as part of the test case specification. Precomputing expected test result is done only for simple test cases and still requires some human intervention. The process of producing predicted output is expensive.

- Comparison-based test oracle
  It follows capture and playback method. The predicted output is captured from earlier execution of a program. The actual output will be compared with the predicted output. It is a simple process and can be implemented for both simple and complex test cases.

- A third approach to producing complex (input, output) pairs.
  In this approach the input is produced from the given output than vice versa.

The judging correctness approach requires a human in judging the test results whereas the third approach is more suitable for producing program input corresponding to a given output. The first and the third approach are suitable only for simple test cases. The comparison-based test oracle approach is most commonly used, and will be used in this thesis.

Figure 1 describes the comparison-based test oracle approach, which includes a test harness program. A test harness (a combination of program and test data to test the application under different scenarios) takes two inputs: (1) the input to the system under test, and (2) the predicted output. The predicted output or behavior is preserved from an earlier execution or captured from the output of a trusted version of the system under test. The system under test gives the actual results. These two results are compared in the test harness; if both the results are the same then the test result output will be pass, otherwise it will be fail.
Richardson et al. [14] have stated that a test oracle contains two parts: oracle information and oracle procedure. Oracle information is used as the expected results and an oracle procedure is used to compare the oracle information with the actual output. Different types of oracles can be obtained by changing the oracle information and using different oracle procedures.

Using test oracles in test automation has become very important since it has been found to make automation testing equal or better than manual testing [13]. Test oracle automation is possible only if there are previous results or standard working version of the system which we want to automate.

Traditional test oracles have two problems:

- Sprenkle et al. [27] have stated that multiple test oracles are needed for each function of web application, which are expensive to develop and it is tedious to maintain multiple oracles.
- Kaner [28] has discussed that the testers usually test only the important criteria (only for basic flows) of a web application, most of the other criteria are neglected since its time consuming to check different criteria, that is they develop test oracles to address only small subset of the criteria which are needed for a web application to function correctly. Web application testing is considered to be completed when it is checked with different combinations of criteria.

In this thesis the focus is on how to use ontologies for test oracles, which solves the above traditional test oracle problems.
3.2 Ontology-based software testing

Pisanelli et al. [15] have discussed that the ontologies of the software systems domain are of great support in avoiding problems and errors during all stages of the software product life cycle. Ontologies have been used in different phases of the software life cycle. Classifications of ontologies are done according to the generality level, type of conceptualization structure, nature of the real-world issue, richness of its internal structure, subject of the conceptualization.

In this thesis, the two steps proposed by Paydar and Kahani [10] for using ontologies in software testing process are followed: a) Developing the ontology which captures the required knowledge to perform the testing process and application domain knowledge. b) To define procedures for using the knowledge embedded in the ontology.

3.2.1 Ontology-based testing approaches

Wang et al. [16], Nguyen et al. [17], and Nasser and MacIsaac [18] have proposed approaches for ontologies based software testing, which are described below.

Wang et al. [16] have demonstrated how ontologies can be used for test case generation and applied for testing web services. The authors address the challenges of automatic test case generation of web services and propose a model driven ontology-based approach with the purpose of improving test formalism and test intelligence. In this approach the authors have used the Petri-Net concept (mathematically defined, and used to provide unambiguous specifications and descriptions of applications) and defined a Petri-Net to provide a formal representation of the web ontology language for web service and have developed Petri-Net ontology for test generation.

Nguyen et al. [17] have proposed a novel approach for automated test case generation using ontologies. The authors have developed the agent interaction ontology and combined with domain ontologies by ontology alignment techniques and integrated the proposed approach into testing framework called e-CAT, which generates and evolves test cases automatically and executes them continuously, resulting in a fully automated testing process, which can run without human intervention for long time.

Nasser and MacIsaac [18] have discussed how ontologies can be used for unit test case generation. The authors have suggested to use ontology reasoning to generate unit test objectives. The authors’

2Available at http://www.petrinets.info/docs/pnstd-4.7.1.pdf
3Available at http://www.aamas-conference.org/Proceedings/aamas08/proceedings/pdf/demo/AAMAS08_demo10.pdf
The objective is to improve unit test case generation by generating test objectives from modifiable test oracles and coverage adequacy criteria specification.

Though ontologies have been used at different levels of testing and in different fields, it has not been used for test oracle automation. In this thesis focus is on how ontologies can be used for test oracle automation for testing web application that is automating the test results verification in testing process.

### 3.3 Web application testing

Web application testing is considered to be more difficult than traditional software application testing because of its characteristics like highly dynamic in nature, a client server architecture and so on [4]. Web application testing uses the same objectives as traditional testing that is web application testing has to rely on test models, testing levels, test strategies, testing processes but because of web applications complexities, traditional testing approaches cannot be used as it is for testing the web application. It has to be modified to the operational environment and new approaches are required.

Pixelbyte lab [19] have described that the web application testing involves:

- **Link testing:** Involves both link checking and testing the link. Link checking is checking whether the link is pointing to the correct URL whereas testing the link is clicking on the link to make sure that the resulting page is correct, and has the required attributes. Link testing also involves checking for any orphan links, whether all the internal and anchored links are working correctly for all the web pages and also involves checking for any broken links.

- **Forms checking:** Forms are used to obtain information from the users. Forms checking consists of fields checking (whether all the fields are valid), checking the work flow between all the forms (the flow should be from first form to second form in case of next and reverse in case of previous). Testing all the fields with the default values, null values, and incorrect values.

- **Testing cookies:** Cookies are small data entities stored on the user’s machine used to maintain login sessions. Testing include checking whether all the cookies are encrypted. Testing the application by switching on/off the cookies in the browser.

- **Testing database:** Test whether any error occurs on modifying or deleting any database functionality. Test whether all the queries execute and update correctly.

Though automation testing is already available for web application, there is need for further improvement in automated test result verification. In this thesis, a procedure is defined to automate the test result verification for web applications by using ontologies and adopted system testing and black box testing strategies to determine the defects in the web application.
Chapter 4

Procedure for an ontology-based oracle comparator for web application testing

This chapter proposes a procedure for ontology-based oracle comparator for testing web application and the chapter is organized as follows. Section 4.1 defines the procedure for ontology-based test oracle. Section 4.2 describes a general architecture for the ontology-based test oracle automation for web application testing. Section 4.3 describes the proposed framework with available tools for storing the web page information into the ontology. Section 4.4 describes the web application which is used for demonstrating the ontology-based test oracle automation. Section 4.5 describes ontology creation for the website. Section 4.6 describes how the two ontologies one with the expected results and the other with the actual results can be compared with each other.

4.1 Procedure for an ontology-based test oracle

The procedure for ontology-based automated test oracle comparator for web application testing is composed of the following steps:

1. Extracting the web pages of the dynamic web application and storing all the web pages in the ontology.
2. Parsing the ontology which was created for the web application into the OWL format.
3. Comparing two ontologies, one with the expected results and the other with the actual results created for the web application.

4.2 General architecture of the proposed framework

The general architecture of the proposed framework and its components are shown in Figure 2. The proposed architecture consists of a Golden Implementation, the Web Application Under Test (WAUT), a customized HTML to OWL convertor, Ontology ER and AR and the comparator, as shown in Figure 2. All the steps shown in Figure 2 can be fully automated. Below, first the components of the architecture are explained, followed by an explanation of the framework.
The components of the architecture are:

- **Test input**: A set of input data for testing is provided to web application to verify the specific functionality. The test input is predefined either stored in an Excel sheet or in a database.

- **Golden Implementation**: Binder [23] has stated that the test oracles use one or more working versions of an existing system to generate expected results, implementing this process is known as golden implementation.

- **Web Application Under Test (WAUT)**: Once the web application is built and deployed it is ready for testing and testing of which will produce the actual results.

- **Customized HTML to OWL convertor**: The manual construction of a domain ontology is a complex, time consuming and hard for the ontologist to develop an accurate and consistent ontology. One way of addressing this problem is by developing a customized tool for converting a web page in HTML to an OWL ontology.

- **Ontologies**: Ontologies will encode the actual and the expected results of web application under test in a machine processable format which can be used for automated test oracle comparison purpose.

- **Comparator**: Compares the two OWL ontologies and checks whether the two ontologies are equivalent or not. If they are not equivalent then it will show the differences.

The architecture of the proposed framework is described below. This includes how the different components work together. The test result (pass/fail) will be achieved by comparing the expected and actual results.
Figure 2. The general architecture of the proposed framework.

A set of test data are provided to golden implementation as an input. As a result it produces the expected result (ER) from the previous working version of the web application or a standard system. A customized HTML to OWL converter takes this as an input and converts it into the OWL format which is stored in an ontology which we will call from now an ontology ER. A same set of test data are provided to WAUT at run time the outcome of which will be the actual result (AR) and converted into OWL format by using the customized HTML to OWL convertor which is stored in the ontology AR.

The golden implementation provides the expected results stored in the ontology ER and the WAUT provides the actual results stored in the ontology AR, these two results are sent to the comparator. The comparator takes the two ontologies that is ontology ER and ontology AR as an argument and compares them to specify whether the two ontologies are identical or different. The differences will be stored in a log file and can be accessed by a tester. If both ontology ER and ontology AR are identical then the test result is set as ‘Passed’ else if there is any deviation between ontology ER and ontology AR then the test result is set as ‘Failed’.
4.3 The proposed framework with OntoBuilder, OntoBuilderParser and OWLDiff

In this section, the general architecture of the framework proposed in section 4.2 is elaborated by using OntoBuilder\(^4\) and OntoBuilder parser\(^5\) for converting HTML web page into an OWL ontology and OWLDiff\(^6\) comparator for comparing the two OWL ontologies.

OntoBuilder: Is portable with various platforms and operating systems. It generates terms by extracting labels and field names from web forms, and recognizes unique relationships among terms and creates ontologies for any websites in an on-the-fly manner [25]. The OntoBuilder works as follows: firstly the URL of the web page for which ontology should be extracted is given as an input to the OntoBuilder; secondly the OntoBuilder parses the HTML page that is all form elements, field values and their labels are identified and produces an ontology.

OntoBuilder parser: The OntoBuilder parser is a plug-in available in OntoBuilder tool, which converts the ontology of web application created by OntoBuilder to an OWL format [25].

OWLDiff comparator: Binder [23] has classified comparators into three general types: system utilities, smart comparator, and application specific comparators. Either readily available system utilities can be used, or a customized application specific comparators, smart comparator can be developed. OWLDiff comparator is a customized application specific comparator which aims at providing the difference between the two OWL ontologies, it takes two ontologies as an argument and checks whether the two ontologies are equivalent or not. If they are not equivalent then it will show the differences. The differences are indicated in red in OWLDiff comparator.

The proposed framework using OntoBuilder, OntoBuilder parser and OWLDiff comparator is as shown in Figure 3. The working of this framework is explained below.

---
\(^4\) Available at http://ie.technion.ac.il/OntoBuilder
\(^5\) Available at http://sourceforge.net
\(^6\) Available at http://sourceforge.net/projects/owldiff/files/
The difference between the general architecture described in section 4.2 and this architecture is a customized HTML to OWL convertor is replaced with OntoBuilder and the OntoBuilder parser and comparator with OWLDiff comparator.

The same set of input data will be provided to the golden implementation (created from previous working version) and WAUT (created at runtime for the web application) these components produce the expected results and the actual results, which will subsequently be sent to OntoBuilder. OntoBuilder converts and stores it in ontology ER and ontology AR. Both the ontology ER and the ontology AR are not in the OWL format, therefore they will be converted to OWL format by OntoBuilder parser. Here the OWLDiff comparator is used for comparing the ontology ER with ontology AR.

The approach described in the beginning of this section for creating an ontology for web application using the freely available OntoBuilder tool has the drawback of creating an ontology only for the main page of the web application but fails for some pages of the web application.

A customized HTML to OWL convertor can be developed for ontology extraction from the website using the reverse engineering approach\(^7\), which can be used to convert HTML to OWL.

Figure 4 shows the architecture proposed in Benslimane et al. [24] the ready-made HTML to OWL convertor.

---

\(^7\) Reverse engineering approach is a process in which a legacy system is analyzed to identify all the system’s components and the relationships between them [24].
Benslimane et al. [24] have described the ontology building framework for the websites. This consists of extraction engine, transformation engine, and migration engine:

- The Extraction Engine analyzes the HTML pages to identify constructs in the form model schema and permits the extraction of a form XML schema, derives the domain semantics by extracting the relational sub-schemas of forms and their dependencies.
The Transformation Engine transforms the relational sub-schemas of forms into conceptual schema based on UML model and translates the modeling language constructs into OWL ontological concepts.

The Migration Engine is responsible for the creation of ontological instances from the relational tuples.

4.4 Web application form

To demonstrate the extraction of the web page information into an ontology and how it is used for test result verification process, the Finnair airline reservation web site is used. This section describes Finnair airline reservation web site.

A web application consists of web pages which are in HTML format and web application testing involves checking all the web pages that is checking forms, field names, links contained in each web page. An easy example will be used to demonstrate how ontology can be created for the web application.

The Finnair web site is used for booking flight tickets. Figure 5 shows the main page of the Finnair website with reservation form selected as default. The main page of Finnair consists of labels like Schedules, Reservations, Award flights, Hotels, Itinerary, Check-In, Departures and arrivals and so on.

The reservation form for the Finnair airlines contain fields Departure city, Destination city, Departure date, Return date, Number of passengers and so on. The radio button fields ‘Return’ or ‘One-way’

---

Available at http://www.finnair.com/
indicates whether user wants to choose one-way journey or return. Departure city is the city from where the user starts his/her journey. Destination city is the city where user wants to travel. Departure date is a date field which takes the date >= current date, which indicates the journey start date. Return date is a date field enabled only if return option selected at the top. It will take date value >= departure date, if return option is selected else departure date >= current date if it is one-way journey. In Number of passengers field user is able to select number of passengers traveling and he/she has an option to select how many of them are adults, children, and infants. The field search option shows the flexible selection dates. From the field class you can select different class options. If you click on continue, a HTTP request will be sent to a server with the provided input and results will be returned.

Figure 6 shows the reservation procedure for airline reservation. The first step in an airline reservation procedure includes entering the departure and the destination city. The second step includes selecting the date when the user wish to travel. The third step will be selecting the flight in which the user wants to travel among the number of flights which are available. And the fourth step will be the price selection from a wide range of flight prices. And the fifth step will be entering the passenger details like gender, first name, last name, contact number and email address. The next step will be payment it will ask option whether you are going to pay by credit card, internet banking and others. Once the payment is made it will send the confirmation email.

Figure 6. Reservation procedure for airlines.
4.5 Ontology creation for Finnair website

In section 4.4 the Finnair web site is described, this section describes how OntoBuilder can be used for creating an ontology corresponding to Finnair web site.

OntoBuilder described in section 4.3 will create ontology of Finnair airline reservation websites. The process of ontology extraction is divided into three phases. First, input is given to the system an HTML page representing a website main page. Second, the HTML page is parsed and all form elements and their labels are identified. Finally, the system produces version of the ontology. Once terms are extracted, OntoBuilder analyzes the relationships among them to identify ontological structures of composition and precedence. Precedence determines the order of terms in the application according to their relative order within a page and among pages.

![Figure 7. Finnair airline reservation form in OntoBuilder](image)

Figure 7 shows the ticket reservation web page form created by OntoBuilder for the Finnair airline reservation. The web page form consists of fields identical to the one provided upon HTTP request for the Finnair airline reservation.
The Ontology generated by OntoBuilder is shown in Figure 8.

![Ontology generated by OntoBuilder for Finnair website](image)

**Figure 8.** Ontology generated by OntoBuilder for Finnair website

The errors that are possible while translating from HTML to OWL are translation errors.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ontology (View Source for full doctype...)>
<ontology name="www.finnair.fi"
  title="Destinations - Finnair" type="full"
>
  <relationship name="is parent of">
    <source>reservation</source>
    <targets>Continue</targets>
  </relationship>
  <relationship name="is parent of">
    <source>reservation</source>
    <targets>Adults</targets>
  </relationship>
  <relationship name="is parent of">
    <source>reservation</source>
    <targets>Adults</targets>
  </relationship>
  <relationship name="is parent of">
    <source>reservation</source>
    <targets>Children (2 – 15 yrs)</targets>
  </relationship>
</ontology>
```

**Figure 9.** A part of the ontology format created by the OntoBuilder
The ontology created by OntoBuilder will be in format has shown in the Figure 9. The ontology created for the Finnair web application by the OntoBuilder should be parsed into the OWL format. This can be achieved by using the OntoBuilder parser or a customized parser can be developed using the OWL parser. The created ontology for the web application has been parsed to the OWL format using the OntoBuilder parser.

When the ontology is created for the Finnair airline reservation web site it can be viewed and edited with the help of Protégé \(^9\) (an ontology editor tool). Figure 10 shows the ontology for the Finnair airline reservation web site in Protégé.

![Figure 10. Finnair airline reservation form in Protégé](image)

The ontology created at runtime for the Finnair airline reservation becomes the actual results. In a similar manner the ontology for the expected result will be created. The only difference between the two is that the ontology created for actual result will be ontology created at run time whereas ontology created for the expected results will be from the previous working versions or from the standard systems. In this way the ontology can be created for the web application. The next section describes how the two ontologies can be compared.

---

\(^9\) Available at http://protege.stanford.edu
4.6 Comparison between two ontologies

This section describes how the ontologies containing actual and expected results are compared. OWLDiff can be used for comparing the two ontologies. It takes two OWL ontologies as an argument and checks whether the two ontologies are equivalent or not.

According to Maedche and Staab [26], the real-world ontologies not only specify their conceptualization by logical structures, but also by reference to terms that are grounded through human natural language to a great extent.

Ivanova [29] describes that the ontology comparison is usually done at four levels:

- Lexical or vocabulary level: Focus on which concepts, instances, facts, etc. have been included in the ontology, and the vocabulary used to represent or identify these concepts. Evaluation on this level involves comparisons with various sources of data concerning the problem domain and techniques such as string similarity measures.
- Taxonomy level: Estimate and evaluate hierarchical is-a relations between concepts in different ontologies.
- Semantic relations level: Evaluate other relations besides is-a. This typically includes measures such as precision and recall.
- Application or context level: Evaluate how the results of the application are affected by the used ontology.

According to Noy, types of mismatches in an ontology comparison can be

- Language-level mismatches: Difference in semantics of ontology language, can be syntax and expressiveness.
- Ontology-level mismatches: Difference in the structure of semantics of the ontology, can be same terms describing different concepts, different terms describing the same concept, different modeling paradigms, different modeling conventions, different levels of granularity, different coverage and others.

Problems associated with ontology comparison are lack of standard terminology, hidden assumptions or undisclosed technical details, and the dearth of evaluation metrics [31].
To demonstrate how the OWLDiff comparator works, the ontology created for Finnair web site (described in section 4.5) is stored as ontology ER. With some manual changes to ontology ER will result in ontology AR. When these two ontologies are given as an argument to the OWLDiff comparator the difference between them are shown in Figure 11, ontology ER is in the left pane and ontology AR is in the right pane. The contents which are not equivalent are displayed in red.

As seen in Figure 11, the class Reservations does not contain the field ‘Check-In’ in ontology ER but ontology AR contains this field. The differences are shown in red. Other errors can be: the Schedules label is missing in the main page of Finnair web site; Reservation form is containing fields of Check-In form; forms having SEARCH button instead of having CONTINUE button; and combo box having different option, that can be Adults instead of Economy and Business.

The errors that are possible while comparing two OWL files are syntactic errors.

Figure 11. OWLDiff comparator showing the difference between expected and actual results.
The comparator validates whether the test case has been ‘Pass/Fail’ by comparing the actual and expected results. Once the comparator evaluates the actual and expected results it will update the status ‘Pass/Fail’ against the test case ran to verify the specific functionality of the system. If the test case status is marked as ‘Failed’ then appropriate log information is updated for the test cases. The log contains both the actual and expected results and it also specifies which part of the test case has been failed. After running all the test cases an analysis report will be generated by the test tool stating which all test cases has been passed or failed.

In this way the test case result can be evaluated automatically with less human intervention using ontology.
Chapter 5

Prototype for the proposed framework

This section describes the prototype of the proposed framework. It contains the sample model of the application to be built and explains how comparison of web application pages should take place during the test verification process using ‘Finnair’ website. As described in section 4.4 the main page of Finnair web site consists of labels like Schedules, Reservations, Award flights, Check-In, Departures and arrivals and so on. The Reservation form consists of several fields for example: Return, One-way, Departure city, Destination city, Departure date, Return date, Number of passengers and buttons like CONTINUE.

These labels and fields are shown in Figure 12 and Figure 13 which represents the user interface. Figure 14 on the other hand, represents the underlying structure of this interface.

Figure 12. Schedule form of Finnair airlines [22]
Figure 13. Hotels form of Finnair airlines [22]

The prototype for the system is explained by performing screen verification for Finnair web site that is checking whether all the fields, labels, button are present which can be verified by comparing two ontologies, ontology $ER$ (with the golden implementation) and ontology $AR$ (with the some faults) for Finnair web site.
Figure 14 represents the two ontologies, *ontology ER* and *ontology AR* of the Finnair web page. The top node represents the root; node 1 represents Schedule form; node 2 – Reservations form; node 3 – Itinerary form; node 4 – Hotels form; node 10 – Departures and arrivals form; Sub-node 1.1 – Return; sub-node 1.2 – One way; sub-node 4.1 – Departure city and sub-node 4.2 – Guest.

Explanation of Figure 14 is as follows. The *ontology ER* contains all the labels present in the main page of Finnair whereas the *ontology AR* contains only first four labels; this type of difference is identified by the OWLDiff comparator.

In the next level of verification,

*ontology ER*
- Node 1 contains sub-node 1.1 and 1.2.
- Node 4 contains sub-node 4.1 and 4.2.

*ontology AR*
- Node 1 contains sub-node 1.1 and 4.2.
- Node 4 contains sub-node 1.2.

In *ontology AR* sub-node 1.2 is missing under node 1; the sub-node 4.1 and 4.2 is missing under node 4. The comparator recognizes these types of difference between *ontology ER* and *AR*, shown in red in Figure 11 and marks the test case as failed because of the differences.
Figure 15 represents the ontology of reservation form which is described in Figure 6 under section 4.4. Form 1 represents Reservation form, Form 2 – Select dates form, Form 3 – Select Flights. The comparator checks whether the correct form is shown after performing certain action on the screen for example: when continue button is clicked the flow should be from form 1 to form 2 and in reverse order that is form 2 to form 1 when back button is clicked. Ontology ER is having the correct flow that is Form 1 -> Form 2 -> Form 3 whereas Ontology AR is having the wrong flow Form 1 -> Form 3 -> Form 2. These types of differences are identified by OWLDiff comparator.

![Ontology ER and AR](image)

**Figure 15.** Ontology ER and AR demonstrating Finnair forms work flow

The verification of test result is performed with pass or fail criteria. The use of ontology improves the verification process by automating it which makes the process easy and simple to perform.
Chapter 6

Evaluation of Proposed Framework

Evaluation is the last phase of this thesis. Since this is a novel approach, it cannot be compared with the previous approach which uses ontologies for test oracles. Therefore the evaluation is based on how ontology-based test oracle solves the traditional test oracle problems and benefits of using ontology-based test oracles.

Traditional test oracles have two problems: (1) multiple test oracles are needed for each function of web application which will be expensive to develop and tedious task to maintain multiple oracles, and (2) testers usually test only the important criteria of web application most of the other criteria are neglected since it is time consuming to check different criteria, that is they develop test oracle to address only small subset of criteria. By using ontologies there is no need of multiple oracles. That is, all basic terms and relations comprising the vocabulary of a domain can be stored in ontology in a machine processable format which in turn solves the first problem. Since all the terms and their relations are stored in an ontology it is easy and reduces the time to check with different combination for test result verification process which solves the second problem of traditional test oracle.

Currently test results verification of web applications are done by comparing two HTML files. In this thesis a procedure is defined for test result verification by comparing two OWL files which as the following advantages over the HTML-HTML comparison:

- Provides semantics
- Is in machine processable format
- Defines relationship between terms and concepts.

Section 6.1 identifies the benefits of the defined procedure and section 6.2 presents the limitations of the procedure.
6.1 Benefits of the proposed framework

The benefits of ontology-based test oracle comparator are identified by analyzing the proposed framework and are mentioned below:

- The use of ontologies helps to store the information in machine processable format by which maximum automation can be achieved. Since the test result verification is automated it reduces the human intervention to a great extent.
- The verification of test results will be performed at each page level that is it can be performed for each small unit of test cases.
- The test results are accurate and reliable since the test result verification process is automated. Because even a most dedicated tester can go wrong this might be prone to errors.
- Once the test result verification is automated it eliminates the human intervention to a great extent which reduces the need of tester hence cost, and time is reduced.
- Repeatable that is the test result can be verified again and again. Since the test result verification is automated.
- The testing can be performed for all the different combination of test input and output. In this way the web applications are delivered with high quality.

6.2 Limitation of the proposed framework

All the test results cannot be evaluated automatically, cases like

- Verifying how user-friendly the web application interface is
- Cannot detect color related issues, that is change in brightness or color and screen appearance in different browsers.

To show how well the proposed procedure may be suited for test oracle automation, it is evaluated by testing the Finnair airline reservation web application using OntoBuilder in section 4.5 and 4.6. From this evaluation gives an indication that for any web application ontology-based test oracle automation can be achieved.
Chapter 7

Discussion and conclusion

This chapter discusses the potential applications for the ontology-based oracle comparator and concludes this thesis and section 7.1 presents the future work.

In today’s world, web application is playing a vital role many organizations carry out their business through web applications. These web applications are frequently changed because of bug or introduction of new functionality so testing them is a difficult task when compared to testing other static applications. So automating the web application testing has become very important which saves time and money. Many automation tools for web application testing have been introduced but automation of test result verification process needs to be improved further. Test result verification process can be automated by using test oracles comparator. In this thesis, the problems associated with current automated oracle comparator for testing web applications have been identified, described in section 3.1.3. This thesis investigates the current works that have used ontology in the software testing process, described in section 3.2 and proposes a novel approach for test result verification process using ontologies, which extracts ontology from web applications, described in section 4.5 and can be applied to a broad range of websites. By comparing two ontology files instead of comparing two HTML files maximum automation can be achieved and human intervention can be reduced to a great extent since ontology provides semantics, defines relationship between terms and concepts and is in machine processable format. The issues when comparing two ontologies can be same term describing different concept, no standard terminology, different modeling conventions, different levels of granularity, described in section 4.6. The errors that are possible while comparing two OWL files can be syntactic errors and sometimes OWLDiff comparator can take long time for comparing and end up with an error message saying comparison failed.

The errors that are possible while translating from HTML to OWL are translation error; sometimes some terms cannot be detected while translating from HTML to OWL this can be prevented by
developing or using the customized HTML to OWL convertor. Apparently the proposed approach is practical and helpful, described in section 4.6.

This thesis addresses the research questions identified in this research by making one important contributions. The contribution is, proposes the general framework for how ontology-based test oracle comparator can be achieved for web application testing, described in section 4.2 and suggestions of how available tools can be used for test result verification process of web applications, described in section 4.5.

7.1 Future work

There are several things in the ontology based test oracle automation that could be further investigated. Firstly, this thesis concentrates on functional system testing and black box testing strategies for the web application testing. Future work may include how the ontology based test oracle automation can be achieved with respect to other levels of testing and other testing strategies. Secondly, it includes the implementation of the defined procedure by developing the integrated tool which should consists of customized HTMLtoOWL convertor, parser and OWL Diff comparator; improvements to the defined procedure are possible. Finally, the framework can be generalized for other context by finding out the different kind of application for which framework needs to be defined and what kind of customized tool is required.
Chapter 8

References


