AUTOMATED TEST CODE GENERATION AND EXECUTION SYSTEM FOR WEB

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

SÜLEYMAN FATİH İŞLER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

JANUARY 2015

AUTOMATED TEST CODE GENERATION AND EXECUTION SYSTEM FOR WEB

Submitted by SÜLEYMAN FATİH İŞLER in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems, Middle East Technical University by,**

Prof. Dr. Nazife Baykal _____
Director, **Informatics Institute**

Prof. Dr. Yasemin Yardımcı Çetin _____
Head of Department, **Information Systems**

Assoc. Prof. Dr. Aysu Betin Can _____
Supervisor, **Information Systems, METU**

Examining Committee Members:

Assoc. Prof. Dr. Pınar Karagöz _____
CENG, METU

Assoc. Prof. Dr. Aysu Betin Can _____
IS, METU

Assist. Prof. Dr. Erhan Eren _____
IS, METU

Assist. Prof. Dr. Tuğba Taşkaya Temizel _____
IS, METU

Assoc. Prof. Dr. Vahid Garousi Yusifoğlu _____
Software Engineering, Atılım University

Date: **20.01.2015**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Süleyman Fatih İşler

Signature : _____

ABSTRACT

AUTOMATED TEST CODE GENERATION AND EXECUTION SYSTEM FOR WEB

İŞLER, Süleyman Fatih
M.S., Department of Information Systems
Supervisor: Assoc. Prof. Dr. Aysu Betin CAN

January 2015, 67 pages

With the development of Web 2.0, the trend in application development has moved from desktop applications towards to web applications. Although there are different ways of testing web applications such as record/replay systems and manual testing, the common practice of web testing is accomplished by manually implementing test script codes from test cases written in software test documents and then run them on test automation tools. Implementation of test script codes is time-consuming process and also requires technical knowledge. To use test automation tools software testers require to have deep knowledge of scripting language and experience of web testing tools. To eliminate technical requirements of web testing tools and enable even non-technical people test web applications, in thesis we propose an automatic web testing tool ATCGES-WEB that automatically generates and runs test scripts by just using page contents and test cases written in English. The proposed tool also reports uncovered DOM elements on pages to software tester in order to be sure that all testable DOM elements are processed.

Keywords: Web Testing, Test Automation, DOM Element Coverage

ÖZ

WEB UYGULAMALARI İÇİN OTOMATİK TEST KODU ÜRETEN VE KOŞAN SİSTEM

İŞLER, Süleyman Fatih
Yüksek Lisans, Bilişim Sistemleri Anabilim Dalı
Tez Yöneticisi: Doç. Dr. Aysu Betin CAN

Ocak 2015, 67 sayfa

Web 2.0 gelişimiyle birlikte uygulama geliştirme modası masaüstü uygulamalardan web uygulamalarına doğru yönelim gösterdi. Web uygulamalarını test etmek için kaydet/oynat ve manuel test etme gibi değişik yöntemler bulunmasına rağmen, web uygulamalarının testinde test senaryolarından manuel bir şekilde test script kodlarının üretilip bu kodların test otomasyon araçlarıyla koşulması yaygın olarak uygulanmaktadır. Test script kodlarının geliştirilmesi zaman alıcı ve teknik bilgiye ihtiyaç duyulan bir süreçtir. Ayrıca, test otomasyon araçlarını kullanabilmek için uygulamayı test eden kişinin scripting dilleri ve test otomasyon araçları hakkında derin bilgiye sahip olması gerekmektedir. Web test araçlarının teknik gereksinimlerini ortadan kaldırmak ve teknik olmayan bir kişinin dahi web uygulamalarını test edebilmesi için, bu tez çalışmasında otomatik web test aracı olan ATCGES-WEB'yi tasarladık. Bu araç sadece web uygulamalarındaki sayfaların içeriklerini ve uygulamanın İngilizce yazılmış test dökümanını kullanarak test script kodları üreterek, üretilen test script kodlarını koşabilir bir yapıya sahiptir. Ayrıca önerilen sistem test script kodları tarafından kapsanmayan sayfa içeriğindeki elemanların raporlanmasını yapmaktadır. Böylelikle, uygulamayı test eden kişi sayfa içerisinde test edilebilir bütün elemanların kapsandığından emin olmaktadır.

Anahtar Kelimeler: Web Test, Test Otomasyonu, DOM Eleman Kapsama

To Begüm Erdem,

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my parent Eyüp İşler and Rukiye İşler and my brother Hakan Murat İşler for providing me an environment in which I could follow my dreams and develop my skills. This thesis could not have been possible without the love, care and support of my family.

I would like to express my gratitude to my beautiful fiancé Begüm Erdem for always be there for me, sharing the happiest and the most stressful moments of my life.

I want to thank my friends Çetin Koca and Fatih Karakuş for their encouragement during the development of this thesis.

I express sincere appreciation to my advisor Assoc. Prof. Dr. Aysu Betin Can for her guidance and great support for my thesis research. She always valued my ideas and endeavors, sometimes even more than me. I would also thank Pınar Karagöz, Vahid Garousi Yusifoğlu, Tuğba Taşkaya Temizel and Erhan Eren for kindly agreeing to be in my thesis committee.

Finally, I would like to thank Kafein Software Inc., the company that I am proud to be a part of, especially to my team leader Mehmet Yaman and team members Aylin Gürkan, Sevan Lalikoğlu, Veysel Peru for their understanding during my studies and being the participants of experiment sessions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GUI	Graphical User Interface
HTML	Hypertext Markup Language
AJAX	Asynchronous JavaScript and XML
SRS	Software Requirement Specification
STD	Software Test Document
SUT	System Under Test
DOM	Document Object Model
IDE	Integrated Development Environment
POS	Part Of Speech
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol
ERP	Event Related Potential
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
PRM	Partner Relationship Management
GSM	Global System for Mobile Communication
BNF	Backus-Naur Form

CHAPTER 1

INTRODUCTION

1.1 Overview and Motivation

By the improvement of web technologies in last decade, web application development has become much more popular than desktop application development. Modern web applications enable users to create sophisticated and highly interactive applications which provide complex and rich GUIs. As Nielsen [1] claimed that web applications were formed by almost 100% static HTML pages in 1995. However, with the improvements on dynamic content management technologies such as AJAX and Javascript, starting from the year 2000 only about 50% of web applications contain static HTML interfaces. Although originally Javascript is designed to handle scripting tasks, it plays a crucial role to add dynamicity on web applications. Javascript has ability to trigger methods to make asynchronous calls to server and reflects the response of server to GUI. By this way, it enables to update partial part of GUI without submitting the whole page content and pages become more dynamic. However, because of sophisticated nature of modern web applications, ensuring the correctness of them becomes more challenging.

Software Testing is an operational procedure to check the correctness of an application. Testing in the development lifecycle of web applications has gained importance due to their sophisticated features. In addition, since web applications are updated frequently due to feature updates, user preference changes or bug fix, before each software delivery it is necessary to perform regression testing on applications to detect whether updates cause any faults or not. Hence, test automation plays an effective role in software testing. Test case generation is the first step in the software testing. Test cases are generally designed by non-developers such as business users or domain experts from SRS (Software Requirement Specification) document. Since most of the time business users and domain experts do not have coding skills to implement test cases, to express instructions in test cases they use a natural language like English. Once the test cases are prepared, applications are tested by following instructions declared in test cases manually or by implementing scripts and execute them with a test framework such as Selenium [2] and Watir [3].

Our motivation in this study is to develop an automated web testing tool that generates and executes test code of web applications by just using test cases written in English. Our aim is to eliminate the technical requirements of web testing tools and enable even non-technical users test web applications.

1.2 Problem Definition and Approach

As Kent claimed that during the system regression tests automated testing creates more accurate results than manual testing [4]. Web application tests may be automated by currently available test frameworks such as Selenium WebDriver [29], Watir [3] or Robot Test Framework [5]. However, it is required to have technical coding background to use these frameworks. Also, since these frameworks force testers to use pre-defined keywords while test code implementation, it is expected from tester to have this knowledge. Even with such framework support, testing remains a time-consuming activity because each test case must be constructed manually.

As Yusifoglu and his colleagues [6] suggest that software testing can be divided into five different tasks which are test-case design, test scripting, test execution, test evaluation and test-result reporting. Based on these tasks, in this thesis study we propose an approach to automate test scripting and test execution tasks in web application testing.

The primary objective of our study is to reduce time-consuming manual tasks in web testing. To accomplish this objective, we propose an automated web testing tool which processes functional instructions written in natural language in software test documents and converts them to meaningful code segments for the usage of an automated test framework Selenium. As a result, we eliminate the technical knowledge requirement of web testing and enable non-technical users to test web applications automatically without implementing any code.

Our secondary objective is to enable software tester to increase the quality of test suite by providing supportive functionalities like reporting uncovered element in system under test (SUT) and mutant generation to evaluate the effectiveness of the test suite.

In order to evaluate the usability and effectiveness of our tool, we performed a user survey and mutation testing. Also we compared ATCGES-WEB against Selenium IDE to detect strong and weak parts of our tool. The evaluation results show that ATCGES-WEB reduces the manual steps in software testing. Also since it provides auxiliary tools, it enables to increase the quality of test suite. As the common thought of people who participated the evaluation of our tool, ATCGES-WEB is an effective and user-friendly web testing tool.

1.3 Summary of Contributions

In this thesis, anautomated web test tool is proposed. Throughout the thesis, the

methodology and implementation details of the approach are described. Also the usability and effectiveness of proposed tool are discussed. The specific contributions of this thesis are as follows:

- An approach of automated test script code generation for web applications by just using application test document is implemented. Unlike existing test frameworks, the proposed tool enables to write test cases in free form language. Also since the generated code is compatible with Selenium framework, code execution can be accomplished independently from proposed tool.

- An XPath Finder tool is implemented to trace elements in the pages of the SUT. This tool enables to query page elements by their XPath values. It also lists all elements in a page and by tracing the elements in this list, user obtains the exact XPath of each element.

- DOM Element Coverage Information is provided to user in 3 categories which are coverage of all DOM Elements (Overall), coverage of elements that are responsible to edit a field or select an option (Editable) in page and coverage of elements that trigger an action or make server calls from application front-end (Event Trigger). By using this information user may improve the quality of test cases.

- 3 types of mutant operators (Type Mixer, Order Shifter and Event Killer) are proposed to evaluate the quality of existing test cases.

- A GUI is provided to user to prepare test document by using tool. Once the preparation of test document is completed, test script code of SUT is also generated. As a result the manual process in code generation is eliminated.

1.4 Organization of the Thesis

The organization of the rest of this thesis is as follows:

- The related work in the field of web test automation is described in Chapter 2. Also the concepts that someone should know to understand our approach are explained.

- Chapter 3 starts with general description of our test automation tool. Then the modules in our tool and the details of each module are explained. Next all phases in test automation process are described in detail.

- In Chapter 4, the techniques that we used during the evaluation of our tool are discussed. Two subject web applications that we performed automated test, are introduced. The comment and grades given by participant are presented.

Also the details of mutation testing and the results are discussed. In addition, a comparison of our tool against Selenium IDE is presented.

- Chapter 5 concludes the thesis by giving the summary of our approach and the results that we gathered from the experiments. Also, possible future works and extensions related with our approach are discussed.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

2.1 Background Knowledge

In this section, we give the details of a set of concepts to clarify our approach on web test automation. These concepts are the explanation of part of speech (POS) Tagging Process, the definition of XPath and the way of element search in page source of SUT, the definition of DOM and the description of Editable and Event Trigger DOM Elements and brief introduction about Selenium Test Framework (Selenium IDE and Selenium WebDriver)

2.1.1 Part Of Speech Tagging

Like most of natural language, in English each word in sentence has special role. Based on position in sentence or behavior that is shown, each word takes over a role in sentence such as noun, verb, adjective and etc. To analyze textual contents and label words in sentences with related tags based on their role in sentence, Part-Of-Speech (POS) Tagger software is used. Since all POS Taggers accept a language rule set in their training, the accuracy of tagging mainly based on the definition of rules in provided set.

Stanford POS Tagger [7] is a commonly used POS Tagger library for language processing tasks. In our proposed system we use Stanford POS Tagger library for extracting information from test documents to generate a test code. The test document defines the steps to perform in each test case. Each test step defines an action (e.g. click) to be performed on a part of the page (e.g. button). Our system uses the tagger library to extract action verbs and nouns from test step definitions.In the following table Table 1, tags that are used by ATCGES-WEB and their abbreviations are illustrated with sample words.

Table 1 Sample Tag Abbreviation and Definitions

POS Tag	Abbreviation	Sample Word
VB	Verb, base form	enter
VBD	Verb, past tense	displayed
VBG	Verb, gerund or present participle	pressing
VBN	Verb, past participle	clicked
VBZ	Verb, 3rd person singular present	leaves
NN	Noun, singular or mass	button, field
NNS	Noun, plural	cars
NNP	Proper noun, plural	Username, Password

Assume that there exists a test step like "Enter Username as 'Fatih' in Registration page." in our test suite. The Stanford POS Tagger tags the corresponding test step as follows,

Original Text:
Enter Username as 'Fatih' in Registration page

Tagged Text:
Enter/VB Username/NNP as/IN 'Fatih'/NNP in/IN Registration/NNP page/NN

By using tagged version of test step, ATCGES-WEB uses VB, VBD, VBG, VBN and VBZ tags to identify action type of test step. In the given sample test step, "Enter" is detected as action. Then by using a reference dictionary, ATCGES-WEB specifies the type of action. The nouns are to be matched to DOM elements in the page. In the given sample, "Username", "Registration" and "page" are tagged with noun specific tags. The details of how ATCGES-WEB processes test steps is explained in Chapter 3.

2.1.2 XPath: The XML Path Language

XML Path Language [8] is a query language defined by the World Wide Web Consortium. It is used for finding elements in XML documents and HTML page sources. To select elements in source document XPath patterns are defined. Once an XPath query is defined, all DOM elements and their attributes in web page are traced to find a match. Although there are many ways to define XPath pattern to select element in page source, in the scope of this study we just focus on order based XPath patterns and attribute based XPath patterns.

Order Based XPath patterns focus on the order of elements among the all element in page. The generic formula of Order Based XPath pattern is shown below.

$$Order\ Based\ XPath\text{: } (//*)[n] \ where\ n\ is\ the\ order\ of\ element\ in\ all\ elements$$

By using "*" letter in expression, it is specified that search is done among the all element in source document. Instead of using "*", by giving valid HTML tags such as input or button, search may be accomplished in smaller set of elements. Order Based XPath queries are vulnerable any modifications in source document. Since the order of elements is updated for each modification in page, the XPath of elements should be updated repeatedly to reflect changes in page source.

Unlike Order Based XPath patterns, Attribute Based XPath patterns use attributes of elements such as id or name, while searching pattern matching. The generic formula of Attribute Based XPath pattern is sampled with given formula below.

$$Attribute\ Based\ XPath\text{: } (//*)[contains(@attribute, 'value')]$$

Since Attribute Based XPath expressions depend on elements' attributes, they do not suffer from modification in source document unless attributes are updated.

2.1.3 Document Object Model

The Document Object Model is a standard that is defined by W3C to represent and interact with elements in HTML page source [9], [21]. Because of its language independent structure, it can be supported by different platforms. In HTML page source, DOM defines the attributes of all HTML elements, the events for all HTML elements and the methods to access elements. In source document, DOM is represented by tree-structure. In Figure 1 and in Figure 2, a sample HTML source code and its corresponding DOM structure is shown respectively.

```
<html>
	<head>
		<title>Sample Title</title>
	</head>
	<body>
		<h1>Sample Header</h1>
		<a href="">Sample Link</a>
	</body>
</html>
```

Figure 1 An HTML Page Code for A Sample Web Page

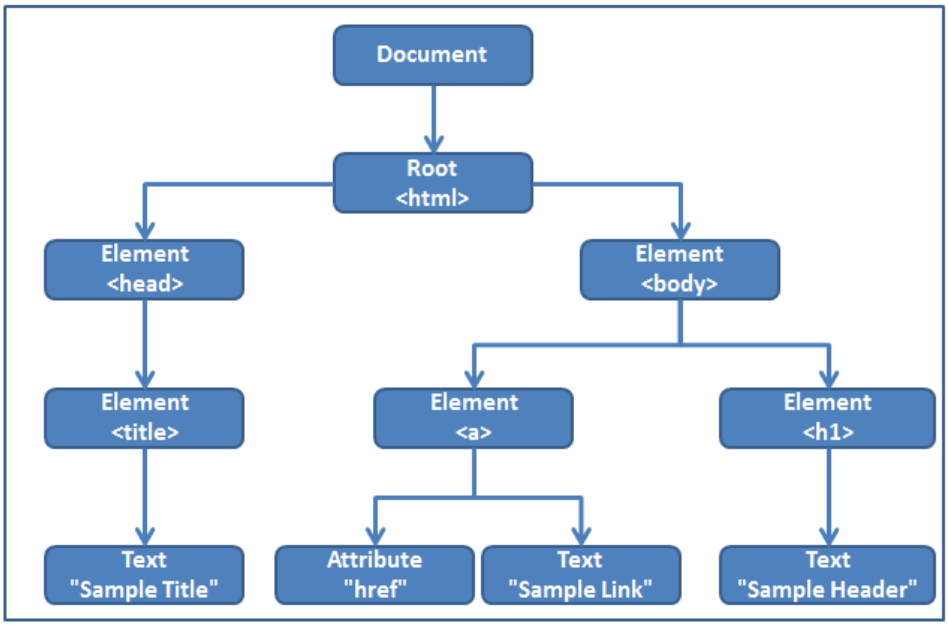

Figure 2 The DOM Element Tree of the Sample HTML Code

In our approach, we classify DOM Elements into 2 different classes such as Editable and Event-Trigger elements based on their characteristics. The DOM Elements that are able to edit a field or select an option on page are counted as Editable DOM Elements. Specifically, the DOM Elements with tag “input”, “select”, “datalist” and “textarea” are located in Editable DOM Element class unless they are not hidden in page. On the other hand, we call DOM Elements as Event-Trigger when elements have ability to trigger event on page. Elements with tag “button” and “a” are accepted as Event-Trigger elements. Apart from “button” and “a“ tag, any DOM Element can also trigger event if it has an event method definition in its attribute list. The complete list of HTML event methods can be found in W3C website [10].

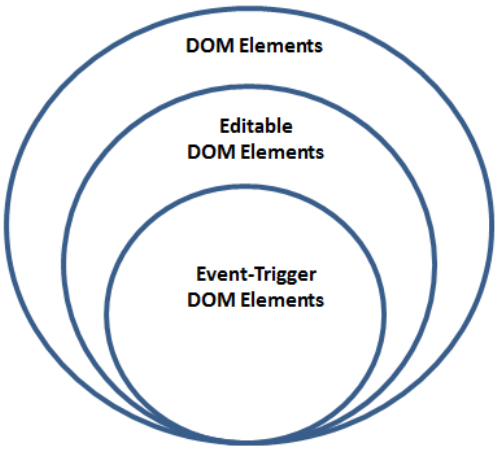

Figure 3 DOM Element Hierarchy

As it is seen hierarchy given in Figure 3, all elements in document are assumed as DOM Element. Underneath the DOM Elements class, we locate Editable DOM Element class to cover editable elements. Since all DOM Elements are able to trigger event by adding HTML event method definition in their attributes list, Event-Trigger DOM Element class is located into the deepest level in hierarchy.

2.1.4 Selenium Test Framework

Selenium is a software test framework for web based applications. By automating browser actions, it enables to run tests without using any manual effort. It provides functionalities for technical and non-technical users. For the users who know how to program, it provides a domain specific language Selenese to implement test cases by coding. For non-technical users, it presents Selenium IDE [2] tool to accomplish test process. Selenium IDE is a record-replay system, that users can record their specific action and then validate these actions. Selenium WebDriver is a server that accepts commands for the browser via HTTP. The code written in Selenese is executed by Selenium WebDriver to automate actions defined in code.

In this thesis study, we used Selenium IDE to compare with ATCGES-WEB. With this comparison, we evaluated the strong and weak part of ATCGES-WEB against Selenium IDE. The details of evaluation are given in Chapter 4.

During the implementation of our proposed tool, we used Protractor framework [27] to automate browser actions. Protractor is another test framework specifically developed for AngularJS [28] applications. AngularJS is an open-source web application framework that enables users to develop dynamic single-page web applications. Since Protractor framework is built on Selenium WebDriver [29], the code generated by our tool can be executed by Selenium WebDriver without any problem.

2.2 Related Work

In this section, studies related with automated web test techniques and test frameworks are presented.

2.2.1 Record/Replay Approach

Record/Replay frameworks enable user to automate test cases by recording the all user actions that are taken in the web page under test. After the completion of recording session, by replaying all actions, web pages may be tested continuously. To write test automation scripts, it is not necessary for users to have any technical knowledge about programming [11][12]. Because of their simplicity, Record/Replay frameworks are attractive for all users regardless of their limitations. The most popular record/replay test frameworks are Selenium IDE [2] and Google Window Tester [13].

Besides of their simplicity, record/replay frameworks have a set of limitations such as consistency in test execution and lack of code reusability. Asynchronicity is a key feature in dynamic web applications. By making asynchronous calls to server, each element in page can change its state in page. It is commonly observed that record/replay frameworks have problems in dynamic state change while replaying the actions [14]. For instance, although in actual scenario the script should wait a dialog to disappear to trigger, it cannot detect waiting dialog on page and since it reaches the wrong element in page, hence the test case fails. In order to solve synchronization problems, ATCGES-WEB provide required waiting operations.

Code reusability is another disadvantage of record/replay test frameworks. While recording user actions, record/replay frameworks creates framework specific code segments. Hence, the generated script codes should be updated when the test framework is changed. To reduce this disadvantage; ATCGES-WEB generates script code which is compatible with Selenium and Protractor framework. We designed ATCGES-WEB in a modular way. Each module in ATCGES-WEB has a specific responsibility such as analysis of test cases, generation of script code and code execution in overall test automation process. By modifying the related modules in tool, ATCGES-WEB may be adapted to similar test frameworks such as Robot Framework [5].

2.2.2 Data-Driven Approach

Unlike Record/Replay approach, in Data-Driven approach the test data (both test inputs and expected outputs) and script code are stored in separated files. Embedding the test data into script code causes dependency problems between code and test data. When the test data is updated, the script code should also be updated. If the script code is long, the maintenance of code may take lots of time. It is also difficult to manage coding issues for non-technical user. Since in Data Driven approach the test data is separated from script code, it is counted as a more mature approach than Record/Replay approach [15]. The benefit of Data Driven Testing is the test data can be designed and created even before the implementation of test code. Also since test data and test code can be developed independently, the test maintenance responsibilities can be divided into different people [16].

Although Data-Driven approach provides users a set of benefits, it also has a drawback. For instance, the initial set-up of test process requires programming skills and management. Although the test data can be prepared by non-technical staffs, at a point it is necessary to get coding support from the technical users to complete the test process [17][18]. Since ATCGES-WEB creates the test script codes of web page under the test from its test document and executes them automatically, ATCGES-WEB removes the programming language knowledge requirement on web-testing and enables even non-developers to test web applications without writing any test script code.

2.2.3 Keyword-Driven Techniques

As in Data-Driven approach, in Keyword-Driven approach the test data and script code is separated from each other. Because of this property, Keyword-Driven approach has all advantages that Data-Driven approach has [19]. By using a set of predefined keywords such as Enter, Click, etc., testers can write tests in a more abstract manner. Because of its simplicity, it is best suited for novice testers. With the help of specified keyword, a complete system test that covers all system functionalities can be built. The most popular Keyword-Driven Test frameworks are HP Quick Test Professional [20] and Robot-Framework [5].

The biggest problem in Keyword-Driven approach is the specification of keywords. Since the specific keywords are special for test framework, in the case of changing test framework all keywords in test data should be updated and related test code should be generated again. ATCGES-WEB also uses a reference keyword dictionary while specifying the action types. If test cases contain an unspecified keyword, ATCGES-WEB cannot detect the action type. However, ATCGES-WEB enables the user to fix action type.

CHAPTER 3

AUTOMATED TEST CODE GENERATION AND EXECUTION

3.1 Methodology

Automated techniques in web testing provide great benefits to companies in today's competitive environment. For instance, since automated techniques minimize the manual tasks in web testing, they enable to create cost effective solutions to companies. Also test automation reduces the possibility of erroneous issues caused by manual tasks. Since existing web test automation solutions require technical background, the users who do not have any development background cannot utilize currently available test automation solutions. The other problematic issue on existing web test automation tools is they do not have any suggestive functionality to direct users to improve the quality of test suite. The more adequate test suites are applied on systems during testing phase, the less bugs are reported by users.

To enable even non-technical users play active roles during the tests of web applications and reduce the manual steps in web testing, we propose a test automation tool ATCGES-WEB (Automated Test Code Generation and Execution System for Web). Besides its automation skill, it provides supportive functionalities (DOM Element Coverage and Mutation Testing) to improve the quality of test suite.

The high level design of ATCGES-WEB is shown in Figure 4. As a standard automation process, ATCGES-WEB takes a test document as input that contains the URL address of web page under test (optional information in file) and the textual definition of test cases, and generates the JavaScript code of test suite that is suitable for Selenium framework. Next, the auto-generated test script is executed with a code execution system integrated in ATCGES-WEB and the pass/fail status of test cases and corresponding failure messages are reported. Also a list of covered and uncovered HTML DOM Elements is listed to user.

During the generation of test script code, ATCGES-WEB requires the DOM Elements in web page under test to be mapped with test step in test suite. Although ATCGES-WEB is able to fulfill element mapping automatically, in the case of absence of URL address definition in test document mapping should be done

manually. ATCGES-WEB XPath Finder is an auxiliary tool that enables user to query and find DOM Element in web pages. Although ATCGES-WEB does not require XPath Finder tool assistance to complete its tasks, for the sake of precise DOM Element mapping it is recommended to get the support of XPath Finder.

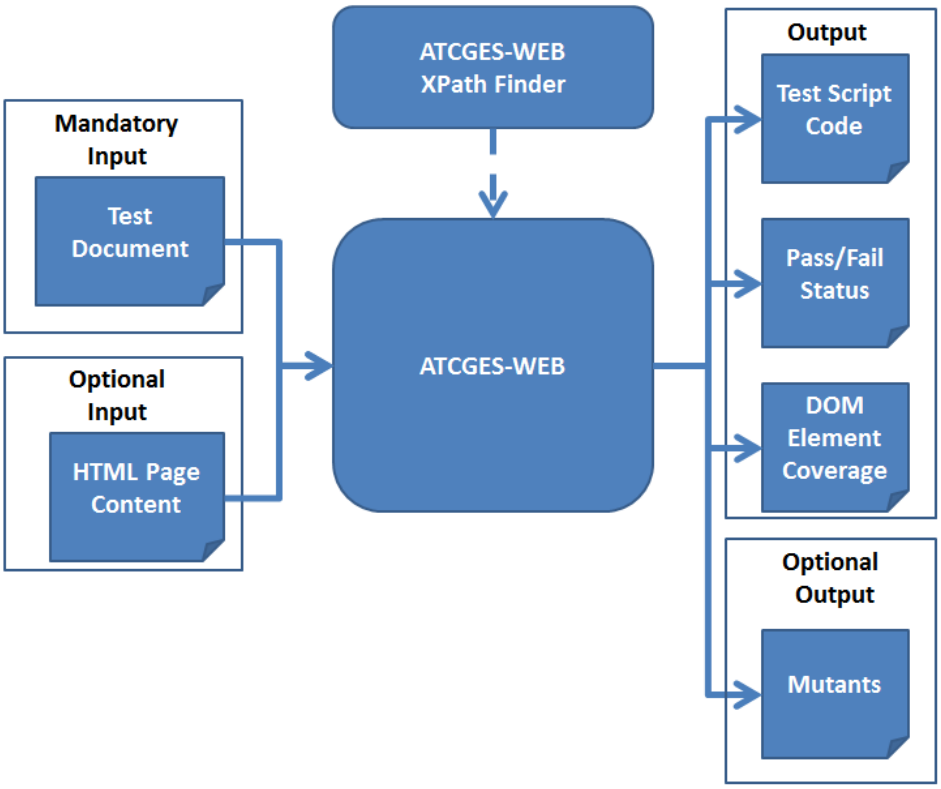

Figure 4 High Level Design of ATCGES-WEB

To check the quality of test suite and improve it, ATCGES-WEB has a functionality to create mutants of open-source web applications. As it can be seen in Figure 4, ATCGES-WEB takes the HTML content of a web page as input. When the HTML Page Content is provided to ATCGES-WEB, by using three mutation operations ATCGES-WEB creates the mutants of web page under test. The detail of mutation operations is given in the following subsections.

3.2 Algorithm

The main algorithm that is working behind ATCGES-WEB system is given at Algorithm 1. The algorithm consists of five fundamental phases. In this section, we give brief information about each phase is given. The details of phases are explained in Section 3.4.

ATCGES-WEB takes a test document that has a specific file structure as the input. The structure of a test document is illustrated in Figure 5.

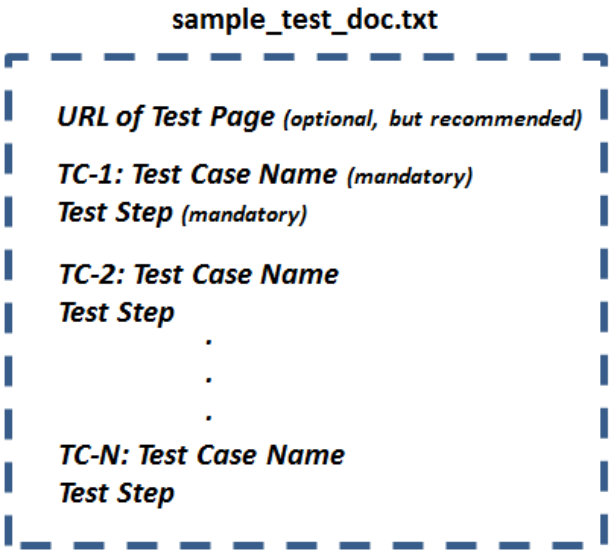

Figure 5 The Structure of Test Document

As it can be seen in Figure 5, there exists a set of mandatory and optional information in a test document. For instance, the definition of URL address of web page under test is optional information in a test document. However, if the URL address is defined, it should be in the first line of test document. Unlike the URL address definition, the name of test case and test steps definition are mandatory information in a test document. Although there does not exist an upper limit for the number of test case definition, it is mandatory to have at least one test case definition in a test document.

In Line 2 in the main algorithm, ATCGES-WEB checks the existence of the page URL address definition in the given test document. If ATCGES-WEB finds an URL address definition, it keeps the URL address for the sake of automated steps related with element mapping.

In Lines 3-5, all test cases are extracted from test document and the relation between each test step and test cases are built.

In Line 6 and Line 7, ATCGES-WEB retrieves the HTML DOM Elements of web page whose address is defined in test document. Once the all DOM Elements are extracted from web page, based on its characteristic each element is partitioned a DOM class such as Editable and Event-Trigger.

In Lines 8-21, firstly each test step in test cases is processed in order to obtain action verb, field candidates and expected value. Action verbs are used for specifying the action type of a test step. Based on action type, automated DOM Element mapping and generated script code show differences. Expected values are the values that are

used for filling fields in web page under test. Field candidates are the set of noun in the test step. To map DOM Elements in web page under test to a test step in document, we use XPath value of DOM Elements. By using field candidates, ATCGES-WEB offers XPath suggestions during DOM Element mapping process. Based on action type, the source set of XPath suggestion is decided. For instance, if a test step has an editable type action, then XPath suggestion is done among the DOM elements that are located in Editable DOM Element class.

```
Input:
1.      tc_doc (textual definition of test cases : string)
Procedure main (tc_doc):
        //Phase 1 – Extraction of Test Cases and Test Steps
2.      page_url = extractPageURL(tc_doc);
3.      test_suite[tc_name] = extractTestCases(tc_doc)
4.      for all test_case tc in test_suite
5.          test_step = extractTestStep(tc)
        // Phase 2 – Partition DOM Elements
6.      dom_elements = extractDOMElements(page_url)
7.      partitionDOMElements(dom_elements, editables, event_triggers)
        // Phase3 – XPath Mapping and DOM Coverage
8.      for all test_case in test_suite
9.          for all test_step in test_case
10.             expectedValue = extractExpectedValue(test_step)
11.             actionType = classifyAction(test_step)
12.             if  page_url is not empty
13.                 if actionType is 'Editable'
14.                     xpath = suggestXPath(editables)
15.                 else if actionType is 'EventTrigger'
16.                     xpath = suggestXPath(event_triggers)
17.                 else
18.                     xpath = suggestXPath(dom_elements)
19.             else
20.                 xpath = manualElementMapping()

21.             visitDOMElement(xpath)
22.     coverage = computeCoverage()
        // Phase 4 – Test Script Code Generation
23.     code = generateTestCode(test_suite)
        // Phase 5 – Code Execution
24.     testResults = executeSuite(code)
```

Algorithm 1 Main Algorithm of ATCGES-WEB

In Line 22, based on XPath definitions used in test steps, covered and uncovered DOM Elements are found.

The processes defined between lines 6-22 show differences depending on the existence of page URL address definition in test document. If there does not exist an URL address definition, ATCGES-WEB will not be able to accomplish DOM

Element partitioning and XPath suggestion. The mapping of the test steps and DOM Element should be done manually by ATCGES-WEB XPath Finder tool.

At Line 23 based on specified action type, XPath mapping and expected value decided in previous steps, ATCGES-WEB generates code for each test step.

As the final process, by automatically executing test script code generated at line 23 pass/fail status of given test suite is reported to user. The system reports a failure message if the DOM element specified with XPath expression cannot be detected during the execution of test code. The system also reports which assertions have failed such as the expected text to be displayed on a label is different what is expected. It is important to note that since the script code that is generated by ATCGES-WEB contains commands compatible with Selenium framework, it can be executed separately by Selenium WebDriver.

3.3 System Architecture

ATCGES-WEB consists of 7 integrated modules which are *Document Parser*, *Test Analyzer*, *DOM Parser*, *XPath Mapper*, *Code Generator*, *Code Executor* and *Mutant Generator*. Each module has a specific task to accomplish while automating the test process. Also there exists an auxiliary tool ATCGES-WEB-XPath Finder to ease XPath mapping process during the test script code generation. The complete system architecture of ATCGES-WEB and dataflow between modules are shown in Figure 6.

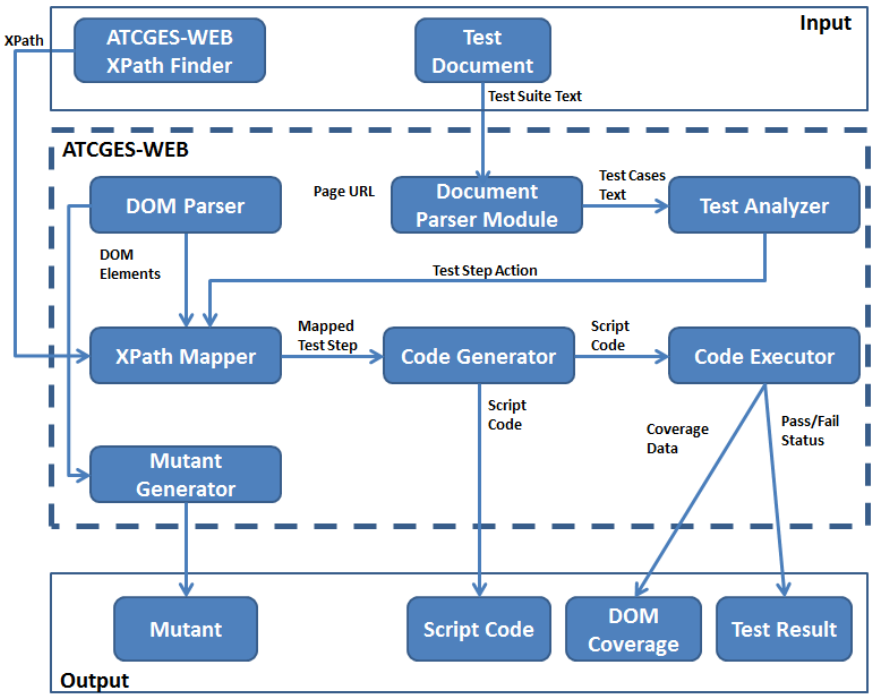

Figure 6 The Overall System Architecture of ATCGES-WEB

ATCGES-WEB takes a test document that contains URL address of web page under the test and the description of test case as input. Firstly, *Document Parser Module* parses the input document to extract descriptions of the test case and URL address of web page under test. Then *Test Analyzer* module processes each test step in individual test cases to extract information for code generation and XPath mapping. Next depending on URL address of web page existence, *DOM Parser* module parses the HTML content of web page to extract DOM Elements. Then extracted DOM Elements are examined to locate editable and event-trigger classes based on element characteristics. The details of DOM Element partition process is described in Section 3.4.2. Then by using classified DOM Elements and preprocessed test steps, *XPath Mapper* module suggests an XPath expression to map current test step with a DOM element in page. It is important to note that URL address of web page is an optional data in test document. In the case absence of test page URL address *DOM Parser* module and *XPath Mapper* module will not be able complete any task and ATCGES-WEB cannot suggest any XPath expression. In such a case, by using *ATCGES-WEB XPath Finder* tool, required XPath expression is obtained manually. It is also possible to validate the correctness of suggested DOM element by *ATCGES-WEB XPath Finder* tool. After the completion of action type determination and XPath mapping *Code Generator* module uses action type and XPath expression to automatically generate the script code of test suite. Finally, *Code Executor* module takes the test script code generated by *Code Generator* module as input and executes it. After the completion of execution, *Code Executor* module reports the pass/fail status of tests and coverage data to user.

As it is shown in Figure 6, there is also *Mutant Generator* module in ATCGES-WEB. *Mutant Generator* module does not have any direct effect on automatic test script code generation or test execution processes. It just creates the mutants of open-source web applications in order to detect uncovered elements by the test cases automatically. To examine the quality of test suite and help users to improve test cases, we decided to develop *Mutant Generator* module in the scope of ATCGES-WEB.

3.4 Definition of Phases

As it is described in Algorithm 1, ATCGES-WEB follows 5 fundamental phases throughout the test automation process. The list of these phases is given below:

- Phase-1: Extraction of Test Cases and Test Steps
- Phase-2: Partition of DOM Elements
- Phase-3: XPath Mapping and DOM Coverage
- Phase-4: Code Generation
- Phase-5: Code Execution

In the following subsections, each one of these phases is described in detail.

3.4.1 Phase-1: Extraction of Test Cases and Test Steps

As it is described in previous section, ATCGES-WEB requires to be fed by a test document to initiate test automation. The test document should obey a set of format rule in order to get full utilization from ATCGES-WEB. The structure of a sample test document and its specification is illustrated in Figure 5.

Test Document contains URL address of the web page under the test and a set of test cases written in English. Definition of URL address is optional but recommended information. If page URL address of web page under the test is provided to ATCGES-WEB, HTML DOM Element related processes such as XPath suggestion and DOM Element Coverage will be handled automatically by the tool itself. Otherwise, these processes should be done manually by user.

The definitions of test cases are mandatory for a test document. There must be a least one test case definition in a test document. The definition of test case must contain a meaningful name that gives brief information about the test case and a set of test steps that will be followed. The generic structure of a test document is illustrated with the following BNF specifications.

Table 2 The BNF of a Test Document

$< test - document >::= < URL >< test - cases > | < test - cases >$

$< test - cases >::= < test - cases > | < test - cases >$

$< test - case >::= < id >$ ":" $< test - case - name >< test - steps >$

$< test - steps >::= < test - steps > | < test - step >$

$< test - step >::=< action - verb >< field - candidates$ """$<expected - value > "'"$" | $< action - verb >< field - candidates >$

$< field - candidates >::= < field - candidates > | < field - candidate >$ |""

A test step must contain an action verb to detect action type and a set of field candidates to map DOM Elements in web page under the test with test step. In briefly, field candidates are the nouns in the test step after the extraction of action verbs and expected values. Expected values are the values that will be entered to an editable filed during a test execution. In order to clarify expected values in test steps, they should be defined between single quotations.

Since the action type is determined based on the verb in test step, each test step must include an action verb. There are 7 types of actions in ATCGES-WEB: *Edit, Event-Trigger, Navigate, Text Validate, Display Validate, Clear* and *Wait*. The details of each action type are explained in Section 3.4.4.1. During the determination of action

type we use a reference dictionary that stores action types and related verbs to define each type. The content of dictionary is given at Table 3.

Table 3 Reference Dictionary

Action Type	Reference Verbs
Editable	enter, leave, select, choose
Event-Trigger	click, trigger, push
Navigate	go to, navigate, visit
Text Validate	verify
Display Validate	validate, display
Clear	clear
Wait	wait

For the test steps that do not contain any verb given in reference dictionary, ATCGES-WEB is unable to detect the action type automatically. However, to handle unidentified action type case, ATCGES-WEB GUI provides necessary functionality to select action type manually. Once the action type is identified, XPath suggestion and test script code generation processes will proceed based on specified action type. The code that is generated and the XPath that is suggested by ATCGES-WEB show differences depending on action type.

When ATCGES-WEB is fed by a test document which obeys rules described above, system extracts the URL address of web page under the test, test cases and test steps from given the test document. Although ATCGES-WEB tool is capable of handling improper test documents, in order to get full utilization from our tool recommended document format should be applied.

3.4.2 Phase-2: Partition of DOM Elements

When there exists a page URL address definition in test document, ATCGES-WEB parses the content of related page to extract its DOM Elements. Once DOM Elements of web page under test is retrieved, they can be partitioned into different classes based in their characteristics. In ATCGES-WEB, 2 types of DOM Element Class are defined which are Editable and Event-Triggers. On the top of these classes, there exists a generic DOM Element class to handle all DOM Elements. As a DOM Element may belong to both Editable and Event-Trigger classes, it is possible to observe that it may not fit any of these classes. For instance, originally all input elements are counted as Editable element. However, input elements can also be Event-Trigger element, if they contain a HTML Event trigger keyword in its attribute list.

Editable DOM Element

Editable DOM Elements are the HTML elements that are to be filled, selected or entered a value. To decide whether a DOM Element is Editable or not, we focus on

its HTML tag. Based on the definition of HTML elements described in W3C [9], we decided to call DOM Elements as Editable Element which have *input*, *select*, *datalist* or *textarea* tag. Sample HTML code snippets using these tags are shown below.

Element With *input* Tag:

```
<input type="text" name="username">
```

Element With *select* Tag:

```
<select>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="mercedes">Mercedes</option>
  <option value="audi">Audi</option>
</select>
```

Element With <*datalist*> Tag:

```
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

Element With <*textarea*> Tag:

```
<textarea rows="4" cols="50"></textarea>
```

Event Trigger DOM Elements

Event Trigger DOM Elements are the ones which have capability of triggering an event on the page. Based on the W3C definition [9], we decided to call DOM Elements as Event Triggers which have *button* and *anchor* type tags.

Element With *button* Tag:

```
<button type="button" id="login_btn">Login</button>
```

Element With *anchor* Tag:

```
<a href="http://www.google.com">Google</a>
```

Independent from the tag type, a DOM Element may still be an Event Trigger DOM Element, if it contains an HTML Event method name in its attribute list such as onclick, onmouseover. The complete set of HTML Event can be found at W3C. Also by defining the type attribute of input element as “submit”, a DOM Element can be also an Event Trigger DOM Element. Event Trigger DOM Elements are referred by action verbs “click”, “trigger” or “push” in test steps.

Element With Event Keyword in Attribute List:

```
<input type="text" onclick="login()">Login</button>
```

Element With Type Attribute:

```
<input type="submit" value="Submit">
```

3.4.3 Phase-3: XPath Mapping and DOM Coverage

After the completion of test case extraction and DOM Elements partitioning, to construct a relation between DOM elements in web page under test and test steps we use XPath expressions. ATCGES-WEB is capable of offering XPath suggestion to test steps automatically. It also enables the users to manage XPath expressions manually.

In Table 4, the definitions of a set of sample test steps and their corresponding XPath expressions are given.

Table 4 Test Step and XPath Mapping

Test Step	Action Verb	XPath	Expected Value
Enter username as 'fatih'	Enter	*//input[contains(@id, "username")]*	fatih
Click Search button	Click	*//input[contains(@id, "search_btn")]*	-
Verify that Search Result Text includes '0 cars found. Page 1, listing results 1-0.'	Verify	*//p[contains(@id, "result")]*	0 cars found. Page 1, listing results 1- 0.
Validate that Error message is displayed	Validate	*//p[contains(@id, "error")]*	true
Clear username	Clear	*//input[contains(@id, "username")]*	-
Wait until dialog is disappeared	Wait	*//*[contains(@id, 'waiting_dialog')]*	-

XPath values are the connection points between elements in page and test steps. These XPath values will be used later in the process while generating the test script code.

In order to get reliable XPath suggestions from our tool, attribute naming of DOM Elements must be done meaningfully. Especially, ATCGES-WEB focuses on id and name attributes of a DOM Element while giving an XPath suggestion. In the following subsection, the automated XPath mapping and manual XPath mapping processes are described.

Automated XPath Mapping: As we described in section 3.4.1, test steps contains field candidates in their structure. Based on field candidates and action types of test steps, an XPath suggestion is done by our tool. While offering XPath suggestion to test step, ATCGES-WEB computes the Levenshtein distance between each noun in field candidates and id or name attributes of DOM Element in web page.

- **Levenshtein Distance:** Levenshtein distance is used for computing the textual distance of two strings [22], [23] and [24]. Since the computation is held by a set of edition operations such as insertion, deletion and update, it is also known as edit distance. By looking at the number of editions done among two subject strings, a distance value is obtained. The less number of edition is done between two strings, the more two strings are textually close to each other. Levenshtein Distance Algorithm is illustrated in Table 5 with a couple of sample.

Table 5 The Levenshtein Distance Illustration

Source	Target	Edit Operations	Distance
Ant	Aunt	insert(‘u’, 2): ‘u’ is inserted at index 2	1
Fatma	Fatih	update(‘m’, ‘i’, 4): ‘m’ is updated with ‘i’ at index 4 update(‘a’, ‘h’, 5): ‘a’ is update with ‘h’ at index 5	2
Samantha	Sam	remove(8): remove letter at index 8 remove(7): remove letter at index 7 remove(6): remove letter at index 6 remove(5): remove letter at index 5 remove(4): remove letter at index 4	5
Oslo	Snow	remove(1): remove letter at index 1 update(‘s’, ‘S’, 1): ‘s’ is updated with ‘S’ at index 1 update(‘l’, ‘n’, 2): ‘l’ is updated with ‘n’ at index 2 insert(‘w’, 4): ‘w’ is inserted at index 4	4

Levenshtein distance algorithm is commonly used in spell checking, speech recognition and plagiarism detection. Because of its popularity on similarity measurement on textual contents, we decided to use Levenshtein distance algorithm while automating the XPath mapping. Since especially non-developer users may not know how to define XPath patterns, by suggesting XPath expressions, ATCGES-WEB reduces the difficulty of defining XPath expressions for users.

Based on action type determined in Phase-1, the number of distance computation can be reduced dramatically. For instance, if test step has an Editable type action, then similarity distance between field candidates and Editable DOM Elements are computed. On the other hand, if test step's action type is Event Trigger, then by using Event Trigger DOM Elements and field candidates in test step similarity distances are computed. If test step's action is neither Editable nor Event Trigger, then Levenshtein distance will be measured for all DOM Elements. By looking at the computed distance, ATCGES-WEB offers the DOM Element that has the shortest distance to noun fields in test step as XPath for the corresponding test step.

Manual XPath Mapping: When the input test document does not contain a page URL address, since ATCGES-WEB cannot find a page to parse, DOM related processes will not be accomplished such as XPath suggestion and DOM Coverage. XPath definitions are crucial source for test script codes and *Code Executor* module. Based on XPath definitions in script code, *Code Executor* attaches the exact DOM elements on page and executes defined commands in code without user interactions. Since XPath definitions have that much important responsibility, in the case of lack of XPath suggestion support, XPath definition of elements in test steps should be mapped manually by the user.

To ease XPath mapping process for users and to remove the ambiguity on DOM element selection, we developed an auxiliary tool that is called as ATCGES-WEB-XPath Finder. It provides a set of functionalities to simplify element selection task in the page under the test. Querying DOM Elements by XPath and tracing each single DOM elements on the page are the fundamental functionalities of this tool. In Figure 7 a snapshot of ATCGES-WEB-XPath Finder tool is given.

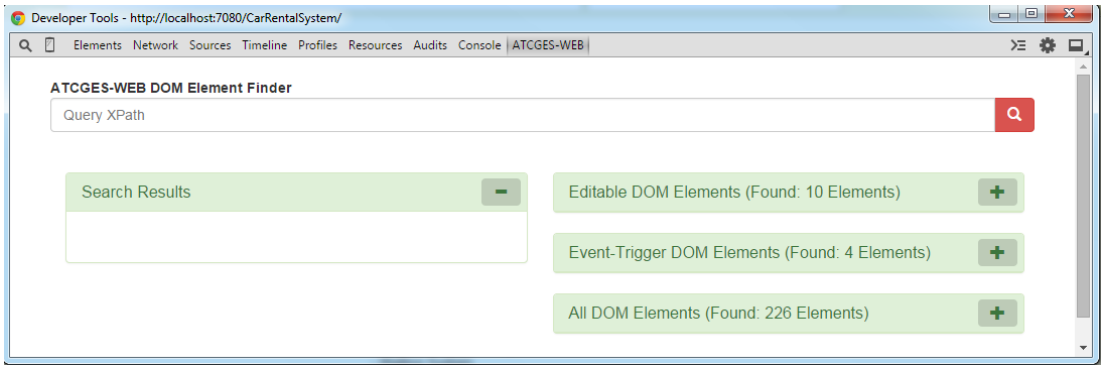

Figure 7 The Snapshot of ATCGES-WEB XPath Finder Tool

ATCGES-WEB-XPath Finder tool is developed as an extension to Google Chrome Browser [25]. For the first time usage, it must be installed on browser. Once it is installed, it may be used on any web page to extract its HTML DOM Element structure. To run ATCGES-WEB-XPath Finder tool, users should press F12 to open Chrome Extension Frame. Then at the top of this frame, ATCGES-WEB tab should be displayed if tool is successfully installed. When ATCGES-WEB tab is selected, given GUI at Figure 7 is shown up.

As it can be seen in Figure 7, there exist five sections which are XPath Query Field, Search Results Panel, Editable DOM Elements Panel, Event-Trigger DOM Elements Panel and All DOM Elements Panel. XPath Query Field enables user to query DOM elements based on their XPath values. Once a valid XPath expression is sent, related DOM Elements are found and listed on Search Results Panel.

When ATCGES-WEB-XPath Finder tool is initiated, all DOM Elements on page is extracted and classified based on their characteristics. Then an order based XPath expression is assigned to each element to identify them. Editable, Event-Trigger and all DOM Elements in page are located in corresponding panels with their order based XPath expressions and HTML values. When the user clicks on an element in one of these lists, the corresponding DOM element on the page is selected and its XPath expression is displayed. For instance, in Figure 8 the first element in Editable DOM Element is the definition of an input field element which is used for specifying a pickup location. When this element is selected from Editable DOM Elements Panel, related element on page is surrounded with a red rectangle to indicate mapping between XPath and DOM Element.

In Figure 8, the selected element's XPath value is (//*)[31]. This is the order based XPath of the element. Although there are several ways to define the XPath expression of a DOM Element, in the scope of ATCGES-WEB and ATCGES-WEB-XPath Finder, we just consider order based XPath expressions and attribute based XPath expressions based on id and name attributes.

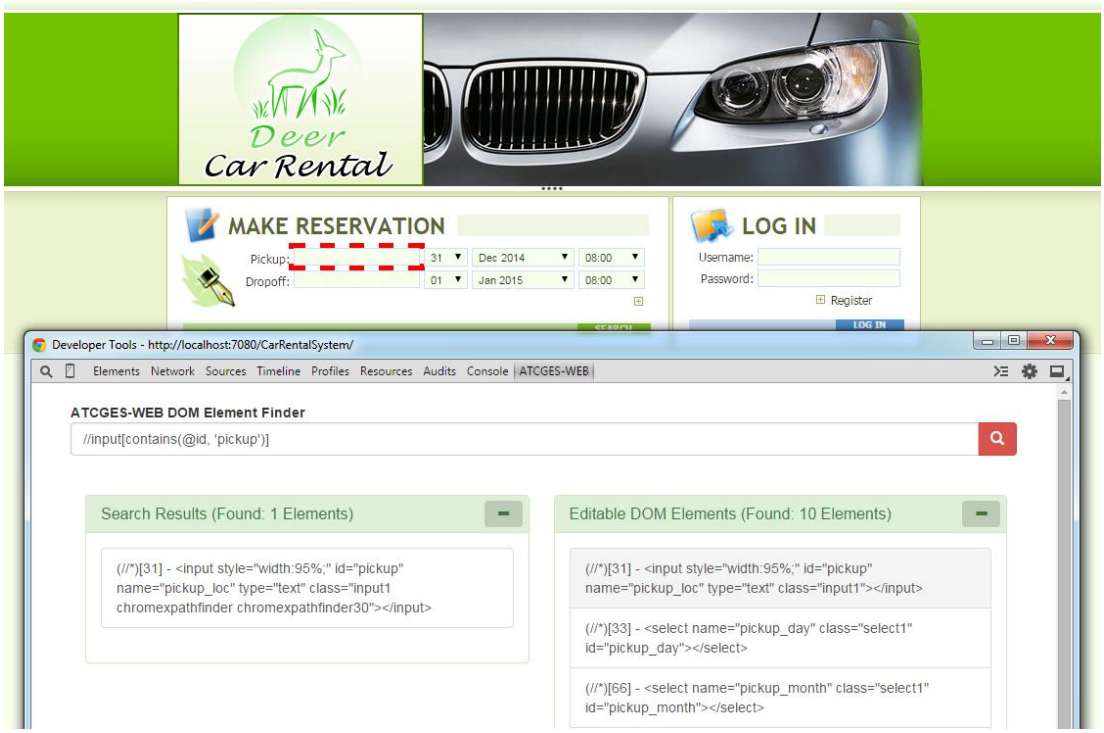

Figure 8 Element Binding on Page

DOM Element Coverage: When an XPath mapping to a test step is performed, ATCGES-WEB checks all DOM Elements that belong to Editable and Event-Trigger DOM Element classes to compute current DOM Element coverage rate. ATCGES-WEB keeps coverage information in 3 categories: Overall Coverage, Editable Coverage and Event-Trigger Coverage. For each update on XPath Mapping, all coverage rates are computed repeatedly and at the end of test code execution, the final rates of DOM Element coverage for 3 categories are reported to user. By using this coverage information, user may extend the scope of test cases to cover more DOM Element in the page.

3.4.4 Phase-4: Automated Code Generation

ATCGES-WEB tool supports 2 types of automatic code generation which are test script code generation and mutant code generation of web applications. Although test script code can be generated for any web applications, mutant code generation is specific for open-source web applications. The details of test script code generation and mutant code generation is described in the following subsections.

3.4.4.1 Test Script Code Generation

Once all the test cases in a test document are analyzed and the XPath mappings are completed, script code is generated automatically. By executing the generated test script code with a script code execution engine such as Selenium or allowing ATCGES-WEB tool execute script code itself; manual steps that should be taken during system test are eliminated.

As it is described in Section 3.4.1, each test step must include an action verb to specify which type of action will be taken by current test step on application page. Based on the type of specified action, ATCGES-WEB assigns the proper commands for the current test step. In ATCGES-WEB, 7 types of action types are defined such as Edit, Event-Trigger, Navigate, Text Validate, Display Validate, Clear and Wait.

Edit actions are the ones that are responsible to send values to input fields or select items from dropdown menus on a web page under test. An Edit Action takes the XPath of DOM Element to be edited and the value to be sent on element as parameter and then edit the specified DOM Element with the defined value. Recall that in reference dictionary, Edit actions are referred by “enter”, “leave”, “select” and “choose” keywords as given in Table 3. As an example, a test step and the corresponding code generated are given in Table 6.

Table 6 A Sample Code for Editable Action Type

Test Step	***Action Verb***	***Suggested XPath***	***Expected Value***
Enter username as ‘fatih’	Enter	*//input[contains(@id, "username")]*	fatih
Generated Code			

browser.driver.findElement(protractor.By.xpath('//input[contains(@id, "username")]')).sendKeys('fatih');

Event-Trigger action is used to trigger an event on web page. It takes an XPath of DOM Element as parameter and triggers the event on element defined with XPath. Recall that in reference dictionary Event-Trigger actions are defined with "click" keyword. As an example, a sample test step and its generated code by ATCGES-WEB are given in Table 7.

Table 7 A Sample Code Segment for Event-Trigger Action Type

Test Step	***Action Verb***	***Suggested XPath***	***Expected Value***
Click Search button	Click	*//input[contains(@id, "search_btn")]*	-
Generated Code *browser.driver.findElement(protractor.By.xpath('//input[contains(@id, "search_btn")]')).click();*			

Navigate action is responsible to visit a web page that is specified with a valid page URL address. Navigate action takes the URL address of page that will be visited as parameter. In reference dictionary Navigation actions are referred by "go to", "navigate" and "visit" keywords. In Table 8, a sample test step and its generated code are given.

Table 8 A Sample Code Segment for Navigation Action Type

Test Step	***Action Verb***	***Suggested XPath***	***Expected Value***
Go to Reservation Page	Go to	-	*http://localhost:7080/CarRentalSystem/reserve.jsp*
Generated Code *browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')*			

ATCGES-WEB provides action types to validate elements' current status on web page. For this purpose it supports 2 types of validation actions which are Text Validate and Display Validate. Text Validate action takes an expected value and XPath of an element to be validated as parameter and compare the text written on the specified DOM element with expected value. In reference dictionary, Text Validate action is referred by "verify" keyword. An auto-generated code segment for Text Validation type action is given in Table 9.

Table 9 A Sample Code Segment for Text Validation Action Type

Test Step	***Action Verb***	***Suggested XPath***	***Expected Value***
Verify that Search Result Text		*//p[contains(@id, "result")]*	'0 cars found.

includes '0 cars found. Page 1, listing results 1- 0.'	Verify	Page 1, listing results 1- 0.
Generated Code *util.validateText(protractor.By.xpath('//p[contains(@id, "result")]'), '*0 cars found. Page 1, listing results 1- 0*')*		

Display Validate action also takes the XPath of element as parameter to detect correct DOM Element in page. However, instead of taking an expected value, it takes a boolean flag to check the visibility of specified DOM Element. In reference dictionary, Display Validate action is defined with "validate" verb. A sample test step that contains a Display Validation action type and its generated code segment by ATCGES-WEB are given in Table 10.

Table 10 A Sample Code Segment for Display Validation Action Type

Test Step	***Action Verb***	***Suggested XPath***	***Expected Value***
Validate that Error message is displayed	Validate	*//p[contains(@id, "error")]*	true
Generated Code *util.validateVisibility(protractor.By.xpath('//p[contains(@id, "error")]'), true)*			

Clear actions are responsible to clean the value typed in editable elements on page. Recall that in reference dictionary, clear actions are defined with 'clear' verb. A sample test step that contains a Clear action type and its generated code by ATCGES-WEB are given in Table 11.

Table 11 A Sample Code Segment for Clean Action Type

Test Step	***Action Verb***	***Suggested XPath***	***Expected Value***
Clear username	Clear	*//input[contains(@id, "username")]*	-
Generated Code *browser.driver.findElement(protractor.By.xpath('//input[contains(@id, "username_")]')).clear();*			

Wait actions are responsible to interrupt the code execution until a specified condition is satisfied. ATCGES-WEB supports 2 types of Wait actions which are *WaitForSeconds* and *WaitForDisappear*. *WaitForSeconds* action type takes a time declaration as parameter during the initialization and stops the execution of test code until the specified time elapsed. A sample test step that contains WaitForSeconds action type and its generated code by ATCGES-WEB are given in Table 12.

Table 12 A Sample Code Segment for WaitForSeconds Action Type

Test Step	***Action Verb***	***Suggested XPath***	***Expected Value***
Wait for '3' seconds	Wait	-	3
Generated Code *util.waitForSeconds(3000);*			

Unlike WaitForSeconds action, WaitForDisappear action type attaches on a specified DOM Element and waits until the corresponding DOM Element is invisible. In reference dictionary, wait actions are referred by 'wait' keywords. As an example, a test step and the corresponding code generated are given in Table 13.

Table 13 A Sample Code Segment for WaitForDisappear Action Type

Test Step	***Action Verb***	***Suggested XPath***	***Expected Value***
Wait until dialog is disappeared	Wait	*//*[contains(@id, 'waiting_dialog')]*	-
Generated Code *util.waitForDisappear('//*[contains(@id, 'waiting_dialog')]*			

3.4.4.2 Mutant Code Generation

Mutation testing is used to detect weak parts of existing test suite by injecting probable faults on source code of applications. In the scope of ATCGES-WEB, we use mutation testing strategy to increase the quality of test suite for open-source web applications. Inspired from Nishiura *et. al* [30], we defined 3 types of mutation operators which are Type Mixer, Order Shifter and Event Killer.

Type Mixer Mutation Operator: Type Mixer Mutation Operators have behavior of changing the type of HTML DOM Elements. There are two types of conversion such as from editable to event-trigger and from event-trigger to editable. Based on the type of DOM elements in original page source, type mixer mutation operator changes them to other type to create mutant code.

- **Conversion Editable DOM Element to Event-Trigger DOM Element**

In ATCGES-WEB, if an element tends to get input values or has ability of selection, these types of HTML elements are assumed to be Editable Elements. To be more specific, elements with HTML tags *input*, *select*,*textarea* and *datalist* are categorized as Editable Elements in page source. While converting an Editable element to Event-Trigger element, the tag of element is changed to button in order to make it an Event-Trigger element. This approach is illustrated with given sample code below.

Original Element:

```
<input name="login_username" id="username" type="text" class="input1" />
```

Mutant Element:

```
<button name="login_username" id="username" type="text" class="input1" />
```

- **Conversion Event-Trigger DOM Element to Editable DOM Element**

In ATCGES-WEB, if an element has the ability to trigger an event, these types of HTML elements are counted as Event-Trigger Elements. For instance, a couple of sample Event-Trigger DOM element and its Type Mixer mutants are given below.

- ***Elements with Tag Button***

Original Code:

```
<button id="login_btn" name="cmd" class="loginBtn" value="LOG IN"
title="login" alt="login" />
```

Mutant Code:

```
<input id="login_btn" name="cmd" class="loginBtn" value="LOG IN"
title="login" alt="login" />
```

In original code although the sample element has "button" tag, in mutant code current element's tag becomes "input".

- ***Input Elements with Type Submit***

Original Code:

```
<input id="login_btn" name="cmd" type="submit" class="loginBtn" value="LOG
IN" title="login" alt="login" />
```

Mutant Code:

```
<input id="login_btn" name="cmd" class="loginBtn" value="LOG IN"
title="login" alt="login" />
```

In original code, sample element's type is shown as "submit". In mutant code the type attribute is removed from element's attribute list.

- ***Elements that Contain DOM Event Keyword(s) in Their Attribute List***

Original Code:

```
<input id="login_btn" name="cmd" class="loginBtn" value="LOG IN"
title="login" alt="login" onkeyup="login()"/>
```

Mutant Code:

```
<input id="login_btn" name="cmd" class="loginBtn" value="LOG IN"
title="login" alt="login" />
```

As it can be seen in the original code, subject element has a DOM event (onkeyup) specification in its attribute list. However, in mutant code this event is removed from element's attribute list. (The complete list of DOM events can be found at [9]).

Order Shifter Mutation Operator: Order Shifter Mutation Operator changes the order HTML DOM elements in page source. Although order of an element in page source does not have any importance when it is queried with attribute based XPath, for order based XPath queries, it has crucial impact.

Original Code:

```
<input name="login_username" id="username" type="text" class="input1" />
```

Order Based XPath: (//*) [167]

Mutant Code:

```
</br><input name="login_username" id="username" type="text" class="input1"
/>
```

Order Based XPath: (//*) [168]

In original page source, username input element is located at 167th element in page. In mutant code, a new line element (</br>) is injected before this element and it becomes the 168th element in page source. After this modification, if someone sends XPath query as "(//*) [167]" to page source, query result will bring new line element instead of input element.

We decided to create order shifter type mutants in order to observe the effects of different XPath query techniques.

Event Killer Mutation Operator: Event Killer Mutation Operatoris specific for Event-Trigger DOM Elements. It breaks the event trigger part of DOM Elements by removing the method which is called by a DOM event. For instance, as it can be seen in the original code below, when the "Logout" link is clicked, a request is sent to server. On the other hand, in the mutated code segment this property is completely removed from the element.

Original Code:

```
<a href="servlet/TestDB?cmd=LOGOUT&requesturl=<%= request.getRequestURI()
%>" title="Logout" id="logout">LOGOUT</a>
```

Mutant Code:

```
<a href="" title="Logout" id="logout">LOGOUT</a>
```

For the following sample code segments, in the original code when user completes to type email address a validation method is triggered to validate the value entered to this field. On the other hand, in the mutant code validation is removed from the element.

Original Code:

```
<input name="email" id="email" type="text" class="input1"
style="width:95%;" maxlength="64" onkeyup="validateEmail()" value="<%=
email %>" />
```

Mutant Code:

```
<input name="email" id="email" type="text" class="input1"
style="width:95%;" maxlength="64" onkeyup="" value="<%= email %>" />
```

3.4.5 Phase-5: Code Execution

As the final step of automation, the script code that is generated by *Code Generator* module is executed by *Code Executor* module and then the result of tests (pass/fail status and the failure messages if any) and DOM Element coverage information is reported to user. On the basis of Code Executor module, we use *Protractor* [26] [27] framework to handle automatic execution of script code.

Protractor is an end-to-end test execution framework which is specifically developed for testing AngularJS applications [28]. Although it provides useful shortcuts for testing of AngularJS applications, by using Protractor framework, it is possible to test any web application. To manage browsers and simulate user actions, Protractor uses Selenium WebDriver [29]. By using Protractor it is possible to navigate to a web page automatically when a valid URL address is provided or click on a button in web page. Protractor also enables to send values to Editable DOM elements and validate something about the application's state such as specific value written on label or visibility of an element in page.

To execute test script code, Protractor requires a configuration file that contains a set of settings about test execution such as Selenium Server set up information, Browser settings, the path definition of test script file and the name of test cases to be executed. A sample configuration file is given in APPENDIX E.

CHAPTER 4

EXPERIMENT AND EVALUATION RESULTS

In order to evaluate effectiveness and usability of ATCGES-WEB, we performed 3 types of evaluation: user evaluation, mutation testing and comparison with a similar tool. During the evaluations, ATCGES-WEB is applied on two characteristically different web applications and by using our tool these two applications' test script codes are generated and executed automatically. In user evaluation, we applied a user survey to an Agile Development Team Members who work on Partner Relationship Management Project of a GSM company. In mutation testing, we randomly injected 3 types of faults in application source code and computed the kill rate of mutants. In addition, we compared our tool with Selenium IDE tool in order to detect advantages and disadvantages of ATCGES-WEB.

In section 4.1, we present the details of the subject applications. In section 4.2, we present the user evaluation. We first explain the approach used during mutation testing and then discuss the result obtained from mutation testing. In Section 4.3, the details of user survey and survey results are introduced. Finally, Section 4.4 the discussion of strong and weak parts of ATCGES-WEB tool alongside Selenium IDE tool.

4.1 Subject Applications

Both Car Rental System and Partner Relationship Management System are the typical samples of dynamic web applications. The first one is a representative of manually generated web application and the second one is a representative of computer generated web application. As most of the modern web applications, they consist of a set of HTML web pages that contain dynamic contents and a backend server layer that manages the requests comes from web pages and creates and sends required responses to web pages. They also contain a database layer (a relational database or object database) to manage data in applications. By the help of AJAX and Javascript function calls, it is possible to make partial updates on web pages without sending whole page content to server which is the typical feature of dynamic web applications. Dynamic web applications have multi-layer architectural structure

and front-end layer (HTML web pages) is just one of these layers. In the scope of this thesis study, we just focused on the test automation of application web pages.Since expression type of HTML DOM Elements is an important feature for the sake of generated test codes' validity, we focused on this characteristic while choosing subject applications. As the subject web applications, we chose a Car Rental System and a Partner Relationship Management (PRM) System. In Car Rental System, all HTML DOM Elements are defined with unique and meaningful id or name attributes. On the other hand, in PRM system all HTML DOM Elements are defined with auto-generated id attributes that are generated on page loading. During the evaluation phase, ATCGES-WEB tool is applied on subject applications and by using our tool; test script codes for these two applications' are generated and executed automatically for selected user scenarios.

While selection of user scenarios, we decided a set of criteria that subject applications have and ATCGES-WEB tool has to handle successfully during test script code generation. The details of used criteria are given in Table 14.

Table 14 The Evaluation Criteria

Difficulty	Form Filling	AJAX calls and Dynamic Content Changes	Page Navigation	Selected User Scenario and Subject Application
Easy	☑			Login User – Car Rental System Login User – PRM System
Moderate	☑	☑		User Registration – Car Rental System Partner Update – PRM System
Hard	☑	☑	☑	Car Reservation – Car Rental System Partner Activation – PRM System

As it can be seen from selection criteria, we define 3 difficulty levels based on user scenario characteristics. On easy level, pages in each subject applications have to have functionality of form filling and form submitting. On moderate level, corresponding pages in the subject applications are required to send asynchronous calls to server and change the content dynamically. For instance, in the case of improper inputs, application shows warning messages on page without user sends a form to the server. In hard level, related pages in subject applications need to include page navigation functionality in addition to other two levels functionality requirements.

4.1.1 Car Rental System

Car Rental System is a very simplified version of web based car rental systems. As

most of the enterprise web applications, in Car Rental System users accomplish a set of user tasks by using system. The fundamental user tasks in a car rental system are car reservation, car rental and vehicle search for specific criteria. Car Rental System is developed as a course project in METU. By using our Car Rental system, users are able to register to system and then by using their username and password specified during registration process, they login to system, search cars to reserve and reserve cars for the specified time periods. For the sake simplicity, while using ATCGES-WEB tool on Car Rental system we ignore most of its functionalities such as external payment system integration, internal messaging system and turning car reservation into rental issue.

Since each HTML DOM element is manually defined with unique id and name attributes in Car Rental system, the generated test script codes by ATCGES-WEB do not suffer from problems related with element order in page. Also in the case of page modifications, test code validity will not be expired unless id or name of HTML element will change. It also enables us to evaluate quality of XPath suggestion that ATCGES-WEB tool provides.

Before we used ATCGES-WEB tool on Car Rental System, we had to prepare a test document. Since Car Rental System does not have an STD (Software Test Document) document, we had to extract test cases of selected scenarios and document them from its SRS (Software Requirement Specification) document in plain English. Then by giving this prepared test cases as input to ATCGES-WEB tool, test script codes are generated.

The written test cases and its test script code generated by ATCGES-WEB tool are provided be seen in APPENDIX A section.

4.1.2 Partner Relationship Management System

As the second subject, we chose the Partner Relationship Management System of well-known GSM company in Turkey. This company gives GSM services to approximately 16.2 million customers and each single year this numbers grows exponentially. This GSM company has a couple of thousands of partner offices spread all province and county into Turkey. In order to manage all of its partners from a common headquarters, this GSM company develops and uses its own Partner Relationship Management (PRM) system. By using the PRM system, users manage partner related tasks such as activating a partner, updating its information, managing staffs in partners, sending/receiving request to headquarter, pricing issues and so on. PRM system is a kind of communication link between a couple of thousands of partners and company HQ. Therefore there are numerous active clients interacting with PRM system everyday. Since PRM application plays a key role in company operations, even a small bug in application can stuck all processes related with partners. Detecting possible faults in PRM application and removing them from the system on time has crucial impact due to its huge number of clients. In the scope of this thesis study, we used ATCGES-WEB tool for login, partner activation and partner information update user scenarios.

The characteristics of expressing DOM elements show difference when we compare it with Car Rental System. All HTML DOM elements in PRM system are defined with auto-generated unique id that is assigned on page loading. Since in every single page load ids are regenerated, it is impossible to guess which id is assigned to single element in page. Because of this characteristic, while using our tool on PRM system, we had to use ordered based XPath expressions to map DOM elements. Order based XPath mapping are vulnerable to page modification. Once a page is modified, its elements' XPath is required to be updated in test script code.

Since test cases of PRM system written in Turkish, as the first step we had to translate them into English for the selected user scenarios. Then by using translated STD document, our tool is used to generate test script codes for the selected user scenarios in PRM system.

For the sake of commercial privacy issues and company specific information contains in PRM system and its documented items, we are not able to share the details of sources and tools that are used during this thesis study.

4.2 Evaluation 1: Mutation Testing

DOM Element Coverage in application is an important parameter to evaluate quality of test suite when testing web applications. As it is mentioned in previous sections, in order to evaluate ATCGES-WEB tool's effectiveness and how well generated code verifies sample application, a set of test cases are prepared for Car Rental System. For the purpose of validating the quality of prepared test cases, we performed mutation testing on test suite and checked DOM Element coverage percentages on subject application.

Type Mixer operator modifies the characteristics of a DOM Element. For instance, Type Mixer operator converts an Editable DOM Element into Event-Trigger DOM Element or vice-versa. Order Shifter operator changes the location of a DOM element in page source. Event Killer operator, as it can be understood from its name, disrupts Event-Trigger DOM Element's event methods and makes them incapable of trigger any action.

4.2.1 Approach

For this evaluation, firstly we created the test script code by feeding ATCGES-WEB with a test document. Then the element mapping is done by id and name based XPaths. The initially reported DOM Element Coverage for the given test suite is 60% on Editable DOM Elements, 50% on Event-Trigger DOM Elements and 57.1% on Overall DOM Elements. The details of DOM Element coverage is given in Table 15.

Table 15 Initial DOM Element Coverage of the Generated Test Scripts

DOM Element Class	# of Covered Element	#of Total Element	Coverage
Overall	8	14	57.1%
Event-Trigger	2	4	50%
Editable	6	10	60%

During the generation of mutants of subject system, we use ATCGES-WEB. To create the mutants of subject system, firstly ATCGES-WEB detects the Editable and Event-Trigger elements on web pages under the test. Then by injecting single fault to page source at a time, mutants of subject system are created automatically based on 3 types of mutation operator. At the end of mutant creation process, we have 23 type mixer mutants, 14 order shifter mutants and 7 event killer mutants of subject system. Then by executing test script code on these mutants, we computed the kill rate of mutant for specified test suite and test script code. Initial kill rates for each mutant type are listed in Table 16.

Table 16 Initial Kill Rates

Mutant Type	# of Killed Mutant	#of Mutant	Kill Rate
Type Mixer	16	23	70%
Order Shifter	0	14	0%
Event Killer	3	7	43%

As it can be seen at Table 16, the most remarkable value is the rate of order shifter type mutants' kill rate. Generated test script code by ATCGES-WEB could not detect them and kill none of order shifter type mutants. The fundamental reason behind this issue is, while the generation of script code, DOM Element mapping is done by id and name based XPaths. Since attribute based XPath definitions are independent from order of DOM Elements in page source, test script code could not detect order changes of DOM Elements. The kill rates of type mixer and event killer mutant are also not significantly enough to claim that given test suite is successful to test subject system.

In order to increase the quality of test suite, we made the following improvements on test case scenarios. To modify test case scenarios, firstly we focused on DOM Elements on live mutants and get the list of uncovered DOM Elements from the tool. Then we added new test steps to the test cases to cover uncovered DOM Elements. After that ATCGGES-WEB generated the script code of modified test suite. Before the generation of script code, besides of attribute based XPath mapping, order based XPath mapping is performed too. The modified version of the test suite and test script code is given at APPENDIX A and APPENDIX C respectively.

To evaluate how test suite and test script code modification affect kill rates of mutant, we executed script code on mutant once more. After test code execution, we

observed that the DOM Element Coverage percentages became to 100% for overall, editable and event-trigger categories and also all mutants were successfully killed.

Mutation Testing showed us that ATCGES-WEB successfully detects the weak parts of test suite. After the detection of the weak parts, users may increase the quality of test suite, by adding new test steps to test cases by using ATCGES-WEB.

4.3 Evaluation 2: User Survey

In order to evaluate the usability of ATCGES-WEB tool, a group of people used our tool on Car Rental System and PRM system. After they used our tool on system, we requested from participants to fill a survey that consists of a couple criteria inspired from Nielsen's Usability Heuristics [31]. The list of selected criteria and brief information about each criterion is given below:

- **Visibility of System Status:** In this criterion participants should evaluate system whether ATCGES-WEB always informs its users about what is going on by giving clear and understandable feedback in reasonable time or not.

- **User Control and Freedom:** In this criterion participants should check that whether ATCGES-WEB tool provides alternative exiting points to user in the case of erroneous issues or not. If so, how effective they are.

- **Error Prevention:** In this criterion participants should look at how well ATCGES-WEB tool prevents erroneous tasks such as a deletion or update operations. For instance, subject system should be able to show confirmation dialogues, warning message and so on.

- **Aesthetic and Minimalist Design:** In this criterion participants should check that how well ATCGES-WEB tool display information and visual components. There must be relevancy between information shown in system and user's current task.

- **Overall Usability and Difficulty:** In this criterion participants should report the easiness and difficulties of using ATCGES-WEB tool and give an overall grade to indicate that ATCGES-WEB tool helps software test life-cycle.

For each criterion we expected from each participant to give a grade from 1 to 5 (1: the worst, 5: the best) and a comment about it. The structure of survey given to all participants is available at APPENDIX D.

4.3.1 Evaluation Setup

ATCGES-WEB was evaluated by agile software development team members who work on PRM and PRM ADMIN project. The team consists of 1 Senior Developer, 1 Senior Tester, 1 Developer and 1 System Analyst. We asked these volunteers, to use

ATCGES-WEB tool on one of the subject system and evaluate our tool in the light of usability criteria given at section 4.3. The participants provide their evaluations by a grade and a comment about each criterion.

The assignment of subject systems is achieved randomly among 4 participants. As the result of this assignment Senior Developer and System Analyst are appointed to Car Rental System and Senior Tester and Developer are appointed to PRM system.

Before the volunteers started to use ATCGES-WEB tool on their assignment system, we gave a set of instructions to them. The details of these instructions are given below.

For the ones who are appointed to use our tool on Car Rental System;

- Since we do not have an STD document of Car Rental system, we prepared test cases for the selected user scenarios by using the requirements in SRS. Then SRS and prepared STD documents of Car Rental system are provided to evaluators.
- Car Rental System is introduced to the evaluators and the fundamental functionalities of system are presented in a 15 minute session.
- ATCGES-WEB and ATCGES-WEB-XPath Finder tools are introduced to evaluators and explained what kind of actions that each tool provides to users. This introduction took 15 minutes.

For the ones who are appointed to use our tool on PRM System;

- For the selected scenarios, related test cases in STD are translated into English and provided to the evaluators.
- Selected scenarios are ran on subject system once, in order to be sure that ATCGES-WEB tool is used on a stable version of PRM system.
- ATCGES-WEB and ATCGES-WEB-XPath Finder tools are introduced to evaluators and explained what kind of actions that each tool provides to users. This introduction took 15 minutes.

4.3.2 Evaluation Results

The comments and grades that are given by system evaluators after user experiment are presented in this section.

Visibility of System Status: System should notify the system status to its user by giving proper information in reasonable time.

Table 17 The User Comments about Visibility of System Status Criterion

Grade	Comment
5	System messages (Process Waiting Dialogues such as Extracting Test Cases, Computing DOM Coverage and Generating Test Code) gives valuable information about system's current status.

5	Confirmation Boxes and Detailed Messages on them provide enough information about what is going to be executed when user confirms action.
5	All test cases in selected test document are listed on UI and the test case that user currently works on is color with red frame. These kinds of facilities keep aware of system status. Also in DOM Element Coverage Page, the current coverage is updated simultaneously when an uncovered element is added any test step.
4	UI does not provide any information about loaded test document. If document name or its path will be shown to user, it can be great

User Control and Freedom: The easiness of exiting an undesired or mistakenly reached state in system. (Undo, Redo)

Table 18 The User Comments about User Control and Freedom Criterion

Grade	Comment
3	Although during Test Step removal system request confirmation from user, once a test step is removed it can't be undone.
3	Tool enables to add new test steps from GUI, it is nice feature. However, newly added test step is added the end of test case. If tool enables user to add test step in any place in test case, tool will be more powerful.
4	When an action is triggered from tool, user waits until process execution completed. To adding the ability of cancellation of triggered processes will provide freedom to reduce an undesired action easily.
4	If a help page that explains input test-case file format or a sample test-case will be added to tool, users will not be confused on system usage.

Error Prevention: Inform user about process to be executed by showing confirmation boxes or similar components before commit.

Table 19 The User Comments about Error Prevention Criterion

Grade	Comment
5	Confirmation Dialogues in tool give required information about actions to be committed.
3	Although tool enables user to create test cases by using UI, once a test case is generated it can't be removed. Also user can't specify the name

	of newly added test case, system automatically assigns a name to it.
5	Tool lists all available operations in action list. If system's current status will not ready to trigger an action, this action type is disabled in list. It is nice feature and prevents user not to commit an action while system is not ready for it. (For instance, it is not possible to trigger code generation process, before loading a test file)
5	While an operation is triggered, system shows an information dialogue about process and until the execution completed, system locks UI in order not to trigger an action by user. That is a nicely thinking feature.

Aesthetic and Minimalist Design: Dialogues and visual components do not give any irrelevant information.

Table 20 The User Comments about Aesthetic and Minimalist Design Criterion

Grade	Comment
5	Using graphical components in system makes tool to look more professional. Especially DOM Coverage Pie Charts and Test Pass/Fail Result Status graphs represent results in practical and understandable way to user.
4	Confirmation and information dialogues do not contain any irrelevant information. However, I detected that when Turkish characters are used in any test step system distort them undesired visual result. If system will be able to solve this encoding problem, it will look more aesthetic.
5	Since available actions are listed in system, user is not confused about the capability of tool. Also action naming is understandable.
5	Using icons on UI enriches understandability of system and makes users to adapt the system easily. Selection of icons in tool is made successfully.

Overall Usability and Difficulty

Table 21 The User Comment about Overall Usability and Difficulty

Grade	Comment
5	Although most of the time system suggests correct XPath suggestions based on element's id and name attributes, it is possible to see that incorrect DOM Element suggestions. In my opinion, XPath Suggestion property needs a bit improvement. However, by using XPath Finder

	tool incorrect suggestions are resolved. In the case of incorrect DOM element suggestions or no suggestion case, I used ATCGES-WEB-XPath Finder tool to query DOM Elements. If someone who has knowledge about how to use XPath, s/he can easily use search facility of XPath Finder tool and as I realized that it works quite reliable if id or name attribute naming is done well. System designer also thought about the users who do not have knowledge of XPath. For these users, XPath Finder tool lists all DOM Elements and their XPath in separated lists. Users can easily trace elements by using this list and realize which element is currently concerned. For all test case scenarios, I successfully create test codes and system executes them without any fail. (Car Rental System Evaluator/Senior Developer)
4	Since elements' ids are assigned automatically at run-time and system gives meaningless auto-generated ids to elements, I couldn't utilize from XPath suggestion facility. All XPath matching are done by ATCGES-WEB-XPath Finder tool and mostly I used order based XPath expressions while using tool. I realized that using order based XPath expressions did not cause any problem and test code is successfully generated and executed. However, designing test code based on order based XPath assignment is highly vulnerable to page modifications. In the case of DOM Element deletion or addition will spoil element's order and generated test code shall be updated in order to reflect XPath modifications. I also realized that ATCGES-WEB tool has some Turkish character encoding problem. However, before I used system since system designer was informed me that system is just supporting english, I do not stress on this problem too much. To conclude, ATCGES-WEB tool can be used to test PRM system. However, for the pages which are finalized and will not be modified in near future. In the case modification, someone shall update test code again. In my opinion, if we will be utilized from ATCGES-WEB tool during regression testing phase, it will make our testing process easier and with less cost. (PRM System Evaluator/Senior Tester)
5	The GUI's in both ATCGES-WEB and ATCGES-WEB-XPath Finder are user-friendly and look very professional. Since I have no knowledge about XPath query language, at the beginning I couldn't understand that whether system suggest correct XPaths or not. So I generated test script codes with automatically suggested XPath and executed tests with these XPath mapping. All test have failed and then I checked the reasons of failures from test results section in tool. At that point, I realized that tests are failed improper XPath mappings. After that, by using XPath Finder tool I updated XPath mappings which caused failure. Since XPath Finder tool represent all elements with their ordered based XPath, I assigned order based XPath to elements by using ATCGES-WEB tool and let the system generate/execute tests with these modifications. All failed scenarios

	were resolved. From my personal perspective if ATCGES-WEB-XPath Finder tool should suggest id based or name based XPath suggestion instead of order based XPath, things can be more understandable. (Car Rental System Evaluator/System Analyst)
4	Test Results section only shows for failure cases. I think success cases should also be presented to user. Also, failure messages of failed test cases are not easy to understand and need improvement. The other drawback of system is, once XPath mapping is done among elements, it can't be saved for future use. Capability of storing this information will make system more practical. The other problem that I encounter is that system facing encoding problem for Turkish characters. (PRM System Evaluator/Developer)

In the light of the user comments, we can claim that ATCGES-WEB is a useful tool for web testing. User accepted that ATCGES-WEB keeps users well-informed of its status by giving proper information in reasonable time. Users were also satisfied from the way of preventing errorneous issues of ATCGES-WEB. In addition users agreed on that the visual components and dialogues are displayed by ATCGES-WEB do not contain any irrelevant information. On the other hand, users reported that ATCGES-WEB should require a set of improvements on user controls such as providing undo feature or cancellation of a process at any time.

From the user comments about overall usability, we can conclude that our proposed tool enables even non-developer users to test web applications automatically. It is also confirmed that once the attribute naming of DOM Elements is done meaningfully, ATCGES-WEB suggests reasonable XPaths. However, users also reported that XPath suggestion feature needs improvements since it sometimes offers incorrect XPaths. To increase the precision of XPath mapping and resolve the wrongly suggested XPath problem, we proposed an auxiliary tool ATCGES-WEB XPath Finder. Users reported that ATCGES-WEB XPath Finder tool helps them to query DOM Elements based on their XPaths. In addition, it is also reported that by looking at the failure reasons messages that ATCGES-WEB reports at the end of each test execution, users were able to fix errors on test steps and reexecute test cases.

To conclude, although our proposed tool requires some improvements on user controls and automated XPath suggestion, it is observed that ATCGES-WEB helps users to automate web application tests. Even non-developer users find our proposed tool is promising on web test automation.

4.4 Evaluation 3: Comparison with Similar Tool

To detect strong and weak part ATCGES-WEB tool, we compared our tool with Selenium IDE. Both system was applied on Car Rental System and used same set of

test cases. During the evaluation of both systems, we followed a set of evaluation criteria such as input types, element assignment, dynamic content handling, test execution, test management, auxiliary test activities and output formats.

4.4.1 Input Types

As input Selenium IDE just takes the URL address of page under the test. Then user follows the test steps written in test document and manually takes the actions on page. While user works on page, Selenium IDE records all his actions. At the end of recording, Selenium IDE replays all recorded actions to simulate user.

ATCGES-WEB tool also takes page URL as Selenium IDE does; however, in addition to the page URL, user should also give test document to our tool as input. When ATCGES-WEB takes the test document as input, it extracts the test cases from input document. Then by processing test steps in each test case a script code is generated and executed automatically. The only manual process in our tool is to fix XPath of elements in the case of doubt on suggested XPath.

4.4.2 Element Assignment

In Selenium IDE, by clicking on element in tested page required mapping with test step and page element is accomplished.

In ATCGES-WEB tool, there exists XPath suggestion for each test step. However, in the case incompatibility on naming of page elements or absence of unique identifier of elements, we observed that system suggest wrong XPath or cannot make any suggestion. In order to reduce improper XPath mapping, ATCGES-WEB-XPath Finder tool should be used. By sending XPath queries to XPath Finder tool, user can assure to map correct mapping between test step and page element.

4.4.3 Dynamic Content Handling

During the comparison of our tool against Selenium IDE, same test suite is used for both systems. (The details of test suite are reported in APPENDIX)

When we executed test cases with Selenium IDE, we realized that Selenium IDE was not able to handle dynamic content changes. For instance, “TC-3: Successful Registration” test case contains step to validate asynchronous processes in registration scenario. In registration scenario while user enters username to the related field, an asynchronous request is sent to server and based on the server response; the label located next to username field shows different text messages. In related test step, the value shown on label is validated. Although the expected value and label value on page are “Username is not available”, Selenium IDE claimed to see “Required” on label and because of unmatched values test case fails.

On the other hand, we did not observe similar issues on ATCGES-WEB tool evaluation. Our tool handled all dynamic content changes successfully.

4.4.4 Test Execution

Although Selenium IDE has different alternatives of test executions such single test step execution, single test case execution and complete test suite execution, ATCGES-WEB tool just supports test suite execution.

4.4.5 Test Management

By using Selenium IDE, a new test step can be added any place in test case. Also it enables multiple test steps and test cases removal as well as single test step removal at a time.

ATCGES-WEB tool just allows single test step removal at a time. As Selenium IDE, ATCGES-WEB enables to add new test steps. However, our tool is only able to add test step at the end of test cases. Arbitrary test step addition is not supported by ATCGES-WEB tool.

4.4.6 Auxiliary Test Activities

ATCGES-WEB tool extracts HTML DOM Element coverage of tested page for given test suite. DOM Element Coverage is an important parameter to improve test suite. By looking at reported uncovered DOM Elements, the scope of test cases can be extended to increase the element coverage which increases the quality of test suite.

ATCGES-WEB tool enables to create mutants of open-source system. By using our tool, 3 types of mutants can be generated and then by evaluating the generated mutants, the scope of test suite can be extended.

Neither DOM Element Coverage nor Mutation Generation facilities are provided by Selenium IDE.

4.4.7 Output Format

In both tool pass/fail status of test cases are exhibited with visual components. Although, in Selenium IDE all test step execution results and failure reasons are reported, in ATCGES-WEB tool just failure reasons are displayed to users.

In ATCGES-WEB tool, at the end of test suite execution javascript test code is generated. The generated test script code can be opened, modified and executed on different machines independent from ATCGES-WEB tool. The only requirement is Selenium Webdriver Manager should be installed on them.

On the other hand, the outputs generated by Selenium IDE can only be opened and modified by tool itself.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, we presented an automated test tool ATCGES-WEB which is specifically developed for test automation of web applications. To test a web application, ATCGES-WEB requires test document of SUT as input. We divided automated test generation and execution process into 5 phases which are *Extraction of Test Cases and Test Step*, *Partition DOM Elements*, *XPath Mapping and DOM Coverage*, *Test Script Code Generation* and *Code Execution*.

In Extraction of Test Cases and Test Step phase, URL address of web page under the test and textual test cases are extracted from test document. Then each test step in test cases is analyzed to extract an action type, a list of candidates and expect value. In Partition DOM Element phase, DOM Elements in web page under the test are classified as Editable and Event-Trigger DOM Elements based on their characteristics. In XPath Mapping phase, each test step is mapped with a DOM Elements in web page under the test. If the URL address of web page under the test is defined in test document, ATCGES-WEB offers an XPath suggestion for test step automatically. Otherwise, user should find the XPath of DOM Element to be mapped with test step manually by using ATCGES-WEB XPath Finder tool. After the completion of mapping all test steps with DOM Elements in page, Test Script Code generation phase begins. In this phase based on action type of test step, XPath mapping done in previous phase and expected value of test step, a code segment is automatically generated. Once the script codes of all test steps are generated, Code Execution and DOM Coverage phase begins. In Code Execution and DOM Coverage phase, by using Protractor test framework ATCGES-WEB executes generated code for test suite and as the result execution it reports to user the pass/fail status of test cases and DOM Element coverage information. ATCGES-WEB also provides user to create mutants of open-source applications. Mutant Creation is an optional process and it does not have any direct effect on test automation. The reason of mutation creation process is to evaluate the quality of test cases.

With regard to test the usability and effectiveness of ATCGES-WEB, we performed a user survey study and mutation testing. In user survey, we requested from a software team members to test two subject web applications by just using ATCGES-

WEB and ATCGES-WEB XPath Finder tools. At the end, we asked each participant to evaluate our tool based on four Nielsen's Heuristics which are Visibility of System Status, User Control and Freedom, Error Prevention and Aesthetic and Minimalist Design. The survey result showed that people find ATCGES-WEB is an effective test automation tool for web applications. To perform mutation testing, three types of mutation operations are defined which are Type Mixer, Order Shifter and Event Killer. Then ATCGES-WEB automatically creates mutants of an open-source web application. The initial mutant kill rate of test code generated by ATCGES-WEB was 70% for Type Mixer, 0% for Order Shifter and 43% for Event Killer. Then by looking at live mutants, we improved the test cases to kill these mutants. After the modification of test suite, we generated the test code and executed it once more by using ATCGES-WEB. As we observed that all mutant were successfully killed.

At the beginning of this study we had a set of objectives which are reducing the time-consuming manual tasks in web testing, eliminating the technical knowledge requirement of web testing and increasing the quality of test suite. From the evaluation results that we obtained, we can conclude that ATCGES-WEB has a positive impact on web test automation. By using our proposed tool, even non-technical users are able to test web applications automatically without writing any script code. In addition, by using ATCGES-WEB, it is observed that the quality of test suite may be increased. Car Rental System and PRM System are two samples of dynamic web applications and the front-end layers of these applications consist of HTML web pages. The results are quite promising for these subject applications. Due to nature of our subject programs, we can generalize these results to three tiered dynamic web applications which contain HTML web pages on their front-end layer and have Javascript and AJAX functions.

A number of interesting extensions and improvements that can be built on the proposed tool are listed in the following list:

- ATCGES-WEB is just able to parse and process test case written in English. As a future work, multi-language support can be added to tool.

- ATCGES-WEB and ATCGES-WEB-XPath Finder work as two separated tools. These two tools should be integrated and published as a Google Chrome Extension to public usage.

- The accuracy of XPath suggestion should be enhanced by using more precise algorithms or hybrid techniques.

- For now our proposed tool can be used just for web application testing. The ability of tool can be extended to test desktop applications as well.

REFERENCES

1. Jakob Nielsen. (1999) *Designing Web Usability: The Practice of Simplicity.* New Riders

2. Gheorghiu, G. *A look at Selenium*. Software Quality Engineering, 7(8):38-44, October 2005

3. Rogers, P., Pettichord, B. & Kohl, J. *Watir: Web Application Testing in Ruby*. Technical report, 2005.

4. Kent John. (2007). *Test Automation: From Record/Playback to Frameworks.* EuroSTAR 2007, Stockholm

5. Bist, S. (2013) *Robot Framework Test Automation*. Birmingham:Packt Publishing

6. Yusifoğlu, V. G., Amannejad, Y., & Can, A. B. (2015). Software test-code engineering: A systematic mapping. *Information and Software Technology*, 58, 123-147.

7. Sari, Y., Hassan, M.F & Zamin, N. *Creating Extraction Pattern by Combining Part of Speech Tagger and Grammatical Parser.* Computer Technology and Development, 2009, ICCTD '09, pages 515-519, Nov. 2009

8. Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F (1998). Extensible markup language (XML). *World Wide Web Consortium Recommendation REC-xml-19980210.*

9. "Document Object Model (DOM) ," [Online]. Last accessed January 4, 2015 from http://www.w3.org/DOM/

10. "World Wide Web Consortium (W3C)," [Online]. Last accessed January 4, 2015 from http://www.w3.org/

11. Ricca, F., Tonella, P. Web testing: a roadmap for the empirical research. In WSE'05: Proceedings of the Seventh IEEE International Symposium on Web Site Evolution, pages 63-70, 2005.

12. Messer's, G. Agile Regression Testing Using Record & Playback. In Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, 353-60. OOPSLA '03. New York, NY, USA: ACM, 2003.

13. "Window Tester Pro User Guide", [Online]. Last accessed January 4, 2015 from https://developers.google.com/java-dev-tools/wintester/html/

14. Dwyer, G., Freeburn, G. Business Object Scenarios: a fifth-generation approach to automated testing- Automating Software Testing (Chapter 22), Addison Wesley Longman, 1999.

15. Bajpai, N. A Keyword Driven Framework for Testing Web Applications. International Journal of Advanced Computer Science & Applications 3, no. 3, 2012.

16. Rice, W. Surviving the top ten challenges of software test automation. In Proceedings of the Software Testing, Analysis & Review Conference (STAR) East 2003. Software Quality Engineering, 2003.

17. Zambelich, K. (1998). Totally Data-Driven Automated Testing. Retrieved April, 4, 2002.

18. Laukkanen, P. Data-Driven and Keyword-Driven Test Automation Frameworks. Helsinki University of Technology, 2006.

19. Kaner, C. Lessons Learned in Sofware Testing: A Context-Driven Approach. John Wiley & Sons, Inc., 2001.

20. Lalwani, T. Kanoujia, S. N., Howarth, T. & Smith, M.(2011). *QuickTest Professional Unplugged.* KnowledgeInbox

21. Marini, J. (2002). *Document Object Model.* Mc Graw-Hill, Inc.

22. Levenshtein, V. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, Vol. 163, No. 4, pp. 845-848, August 1965.

23. Calabi, L. On the Computation of Levenshtein's Distances. TN-9-0030 Parke Math. Labs., Inc, Carlisle (1967), p. 664-675

24. Kouylekov, M., & Magnini, B. (2005). Recognizing Textual Entailment with Tree Edit Distance. In Proceedings of the PASCAL RTE Challenge (pp.17-20).

25. "Chrome Web Store," [Online]. Last accessed January 4, 2015 from https://chrome.google.com/webstore/category/extensions

26. "E2E Testing," [Online]. Last accessed January 4, 2015 from https://docs.angularjs.org/guide/e2e-testing

27. "Protractor end to end testing for AngularJS," [Online]. Last accessed January 4, 2015 from http://angular.github.io/protractor/#/

28. Darwin, P.B., Kozlowski, P.(2013) *Mastering Web Application Development with AngularJS.* Birmingham:Packt Publishing

29. Burns, A., Kornstadt, A. & Wichmann, D. (2009). Web Application Test with Selenium. Software, IEEE, 26(5), 88-91.

30. Nishiura, K., Washizaki, H. & Honiden, S. Mutation Analysis for Javascript Web Application Testing. In Proc. Int'l Conf. on Software Engineering and Knowledge Engineering (SEKE), pages 159-165, June 2013.

31. Nielsen, J (1990). Heuristic evaluation of user interfaces, in CHI '90: Processing of the SIGCHI conference on Human factors in computing systems', ACM Press, New York, NY, pg. 249-256

APPENDICES

APPENDIX A: CAR RENTAL SYSTEM TEST CASES

Test Suite Before Mutation Testing	*Test Suite After Mutation Testing*
http://localhost:7080/CarRentalSystem/reserve.jsp **TC-1: Invalid Pickup Day (Past Date)** Go to Reservation Page Choose pickup_day as '1' Click Search button Verify that Search Result Text includes '0 cars found. Page 1, listing results 1- 0.'	http://localhost:7080/CarRentalSystem/reserve.jsp **TC-1: Invalid Pickup Day (Past Date)** Go to Reservation Page Choose pickup_day as '1' Choose pickup_month as 'Dec 2014' Click Search button Verify that Search Result Text includes '0 cars found. Page 1, listing results 1- 0.'
TC-2: Invalid Dropoff Time (Earlier Time from Pickup Day) Go to Reservation Page Choose pickup_day as '30' Choose dropoff_day as '17' Click Search button Verify that Search Result Text includes '0 cars found. Page 1, listing results 1- 0.'	**TC-2: Invalid Dropoff Time (Earlier Time from Pickup Day)** Go to Reservation Page Choose pickup_day as '30' Choose pickup_month as 'Dec 2014' Choose dropoff_day as '17' Choose dropoff_month as 'Dec 2014' Click Search button Verify that Search Result Text includes '0 cars found. Page 1, listing results 1- 0.'
TC-3: Successful Registration Go to Registration Page Enter username as 'fatih' Verify that username_validation label is 'Username is not available' Clear username Enter username as 'e190424' Verify that username_validation label is 'ok' Enter password as '123456'	**TC-3: Successful Registration** Click Registration Button Enter username as 'fatih' Verify that username_validation label is 'Username is not available' Clear username Enter username as 'e190424' Enter password as '123456' Enter confirm_password as '123456' Enter fullname as 'Fatih Isler'

Enter confirm_password as '123456' Enter fullname as 'Fatih Isler' Enter email as 's.fatih.isler@gmail.com' Click Register button	Enter email as 's.fatih.isler' Verify that email_validation is 'Required' Clear email Enter email as 's.fatih.isler@gmail.com' Verify that email_validation is 'ok' Enter occupation as 'Software Engineer' Click Gender as Male Enter birth_date as '24' Enter birth_month as '08' Enter birth_year as '1986' Click Register button
TC-4: Reservation Without Login Go to Reservation Page Click Search button Verify that Search Result Text includes '9 cars found. Page 1, listing results 1- 9.' Click Second Car in result list Validate that Error message is displayed Enter username as 'e190424' Enter password as '123456' Click Login button Validate that Error message is disappeared	**TC-4: Reservation Without Login** Go to Reservation Page Click Search button Verify that Search Result Text includes '9 cars found. Page 1, listing results 1- 9.' Click Second Car in result list Validate that Error message is displayed Enter username as 'e190424' Enter password as '123456' Click Login button Validate that Error message is disappeared
TC-5: Pickup and Dropoff Specification Go to Reservation Page Enter Pickup Location as 'Ankara' Click on Search button Verify that Search Result Text includes '4 cars found. Page 1, listing results 1- 4.' Enter Pickup Location as 'Istanbul' Click on Search button Verify that Search Result Text includes '5 cars found. Page 1, listing results 1- 5.' Enter Pickup Location as 'Izmir' Click on Search button Verify that Search Result Text includes '9 cars found. Page 1, listing results 1- 9.' Enter Pickup Location as 'Ankara' Enter Dropoff Location as 'Istanbul' Click on Search button	**TC-5: Pickup and Dropoff Specification** Go to Reservation Page Enter Pickup Location as 'Ankara' Click on Search button Verify that Search Result Text includes '4 cars found. Page 1, listing results 1- 4.' Enter Pickup Location as 'Istanbul' Click on Search button Verify that Search Result Text includes '5 cars found. Page 1, listing results 1- 5.' Enter Pickup Location as 'Izmir' Click on Search button Verify that Search Result Text includes '9 cars found. Page 1, listing results 1- 9.' Enter Pickup Location as 'Ankara' Enter Dropoff Location as 'Istanbul' Enter Pickup_hour as '08:00'

Verify that Search Result Text includes '4 cars found. Page 1, listing results 1- 4.'	Enter Dropoff_hour as '08:00' Click on Search button Verify that Search Result Text includes '4 cars found. Page 1, listing results 1- 4.'
TC-6: Sucessful Reservation Go to Reservation Page Click on Search Button Verify that first car in result list is 'BMW 3 Series (2006)' Click first car in result list Click Confirmation button Go to My Reservation Page Validate that BMW 3 Series is in list Verify that BMW 3 Series reservation status is 'Active'	**TC-6: Sucessful Reservation** Go to Reservation Page Click on Search Button Verify that first car in result list is 'BMW 3 Series (2006)' Click first car in result list Click Confirmation button Click on My Reservation Link Validate that BMW 3 Series is in list Verify that BMW 3 Series reservation status is 'Active'

APPENDIX B: SCRIPT CODE BEFORE MUTATION TESTING

```
var util = require('./util/util')

describe('Test_Suite_12_27_2014_18_50_25', function() {
	it('TC-1: Invalid Pickup Day (Past Date)', function() {
		browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')

	browser.driver.findElement(protractor.By.xpath('//select[contains(@name,
		"pickup_day")]')).sendKeys('1');
		browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
		"search_btn")]')).click();
		util.retrieveText(protractor.By.xpath('//p[contains(@id,
		"result")]')).then(function(response) {
			if (response != undefined && response.length > 0) {
				expect(response).toEqual('0 cars found. Page 1, listing
				results 1- 0.');
			} else {
				util.retrieveTextForInput(protractor.By.xpath(
				'//p[contains(@id,"result")]')).then(function(response) {
					expect(response).toEqual('0 cars found. Page 1,
					listing results 1- 0.');
				});
			}
		});
	});

	it('TC-2: Invalid Dropoff Time (Earlier Time from Pickup Day)', function() {
		browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')

	browser.driver.findElement(protractor.By.xpath('//select[contains(@name,
		"pickup_day")]')).sendKeys('30');

	browser.driver.findElement(protractor.By.xpath('//select[contains(@name,
		"dropoff_day")]')).sendKeys('17');
		browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
		"search_btn")]')).click();
```

```
util.retrieveText(protractor.By.xpath('//p[contains(@id,
                "result")]')).then(function(response) {
                        if (response != undefined && response.length > 0) {
                                expect(response).toEqual('0 cars found. Page 1, listing
                                results 1- 0.');
                        } else {

                                util.retrieveTextForInput(protractor.By.xpath('//p[contai
                                ns(@id, "result")]')).then(function(response) {
                                        expect(response).toEqual('0 cars found. Page 1,
                                        listing results 1- 0.');
                                });
                        }
                });
        });

        it('TC-3: Successful Registration', function() {

        browser.driver.get('http://localhost:7080/CarRentalSystem/register.jsp')
                browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
                "username_")]')).sendKeys('fatih');
                util.retrieveText(protractor.By.xpath('//p[contains(@id,
                "username_val")]')).then(function(response) {
                        if (response != undefined && response.length > 0) {
                                expect(response).toEqual('Username is not available');
                        } else {

                                util.retrieveTextForInput(protractor.By.xpath(
                                '//p[contains(@id,"username_val")]')).then(function(
                                response) {
                                        expect(response).toEqual('Username is not
                                        available');
                                });
                        }
                });

                browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
                "username_")]')).clear();
                browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
                "username_")]')).sendKeys('e190424');
                util.retrieveText(protractor.By.xpath('//p[contains(@id,
                "username_val")]')).then(function(response) {
                        if (response != undefined && response.length > 0) {
                                expect(response).toEqual('ok');
                        } else {
                                util.retrieveTextForInput(protractor.By.xpath(
                                '//p[contains(@id, "username_val")]')).then(function(
                                response) {
                                        expect(response).toEqual('ok');
                                });
                        }
                });

                browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
```

```
        "password_")]')).sendKeys('123456');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "confirm_pass")]')).sendKeys('123456');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "fullname")]')).sendKeys('Fatih Isler');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "email")]')).sendKeys('s.fatih.isler@gmail.com');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "register")]')).click();
});

it('TC-4: Reservation Without Login', function() {
        browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "search_btn")]')).click();
        util.retrieveText(protractor.By.xpath('//p[contains(@id,
        "result")]')).then(function(response) {
                if (response != undefined && response.length > 0) {
                        expect(response).toEqual('9 cars found. Page 1, listing
                        results 1- 9.');
                } else {
                        util.retrieveTextForInput(protractor.By.xpath(
                        '//p[contains(@id, "result")]')).then(function(response)
                {
                                expect(response).toEqual('9 cars found. Page 1,
                                listing results 1- 9.');
                        });
                }
        });

        browser.driver.findElement(protractor.By.xpath('//*[contains(@id,
        "car_11")]')).click();
        util.checkVisibility(protractor.By.xpath('//p[contains(@id,
        "error")]')).then(function(response) {
                expect(response).toEqual(true);
        });

        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "username")]')).sendKeys('e190424');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "password")]')).sendKeys('123456');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "login_btn")]')).click();
        util.checkVisibility(protractor.By.xpath('//p[contains(@id,
        "error")]')).then(function(response) {
                expect(response).toEqual(false);
        });
});

it('TC-5: Pickup and Dropoff Specification', function() {
        browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "pickup")]')).sendKeys('Ankara');
```

```
browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
"search_btn")]')).click();
util.retrieveText(protractor.By.xpath('//p[contains(@id,
"result")]')).then(function(response) {
	if (response != undefined && response.length > 0) {
		expect(response).toEqual('4 cars found. Page 1, listing
		results 1- 4.');
	} else {
		util.retrieveTextForInput(protractor.By.xpath(
		'//p[contains(@id, "result")]')).then(function(response)
	{
			expect(response).toEqual('4 cars found. Page 1,
			listing results 1- 4.');
		});
	}
});

browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
"pickup")]')).sendKeys('Istanbul');
browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
"search_btn")]')).click();
util.retrieveText(protractor.By.xpath('//p[contains(@id,
"result")]')).then(function(response) {
	if (response != undefined && response.length > 0) {
		expect(response).toEqual('5 cars found. Page 1, listing
		results 1- 5.');
	} else {
		util.retrieveTextForInput(protractor.By.xpath(
		'//p[contains(@id, "result")]')).then(function(response)
	{
			expect(response).toEqual('5 cars found. Page 1,
			listing results 1- 5.');
		});
	}
});

browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
"pickup")]')).sendKeys('Izmir');
browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
"search_btn")]')).click();
util.retrieveText(protractor.By.xpath('//p[contains(@id,
"result")]')).then(function(response) {
	if (response != undefined && response.length > 0) {
		expect(response).toEqual('9 cars found. Page 1, listing
		results 1- 9.');
	} else {
		util.retrieveTextForInput(protractor.By.xpath(
		'//p[contains(@id, "result")]')).then(function(response)
	{
			expect(response).toEqual('9 cars found. Page 1,
			listing results 1- 9.');
		});
	}
});
```

```
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "pickup")]')).sendKeys('Ankara');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "dropoff")]')).sendKeys('Istanbul');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "search_btn")]')).click();

        util.retrieveText(protractor.By.xpath('//p[contains(@id,
        "result")]')).then(function(response) {
               if (response != undefined && response.length > 0) {
                      expect(response).toEqual('4 cars found. Page 1, listing
                      results 1- 4.');
               } else {
                      util.retrieveTextForInput(protractor.By.xpath(
                      '//p[contains(@id, "result")]')).then(function(response)
               {
                             expect(response).toEqual('4 cars found. Page 1,
                             listing results 1- 4.');
                      });
               }
        });
});

it('TC-6: Sucessful Reservation', function() {
        browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "search_btn")]')).click();
        util.retrieveText(protractor.By.xpath('//*[contains(@id,
        "car_01_info")]')).then(function(response) {
               if (response != undefined && response.length > 0) {
                      expect(response).toEqual('BMW 3 Series (2006)');
               } else {
                      util.retrieveTextForInput(protractor.By.xpath(
                      '//*[contains(@id, "car_01_info")]')).then(function(
                      response) {
                             expect(response).toEqual('BMW 3 Series (2006)');
                      });
               }
        });

        browser.driver.findElement(protractor.By.xpath('//*[contains(@id,
        "car_01_info")]')).click();
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "confirm")]')).click();
        browser.driver.get('http://localhost:7080/CarRentalSystem/
        contracts.jsp')

        util.checkVisibility(protractor.By.xpath('//p[contains(@id,
        "status")]')).then(function(response) {
                expect(response).toEqual(true);
        });

        util.retrieveText(protractor.By.xpath(
```

```
                '//p[contains(@id, "status")]')).then(function(response) {
                        if (response != undefined && response.length > 0) {
                                expect(response).toEqual('Active');
                        } else {
                                util.retrieveTextForInput(protractor.By.xpath(
                                '//p[contains(@id, "status")]')).then(function(
                                response) {
                                        expect(response).toEqual('Active');
                                });
                        }
                });
        });
});
```

APPENDIX C: SCRIPT CODE AFTER MUTATION TESTING

```
var util = require('./util/util')

describe('Test_Suite_12_27_2014_19_36_04', function() {
	it('TC-1: Invalid Pickup Day (Past Date)', function() {
		browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')

	browser.driver.findElement(protractor.By.xpath('//select[contains(@name,
		"pickup_day")]')).sendKeys('1');

	browser.driver.findElement(protractor.By.xpath('//select[contains(@name,
		"pickup_month")]')).sendKeys('Dec 2014');
		browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
		"search_btn")]')).click();
		util.retrieveText(protractor.By.xpath('//p[contains(@id,
		"result")]')).then(function(response) {
			if (response != undefined && response.length > 0) {
				expect(response).toEqual('0 cars found. Page 1, listing
				results 1- 0.');
			} else {
			util.retrieveTextForInput(protractor.By.xpath(
			'//p[contains(@id, "result")]')).then(function(response) {
					expect(response).toEqual('0 cars found. Page 1,
					listing results 1- 0.');
				});
			}
		});
	});

	it('TC-2: Invalid Dropoff Time (Earlier Time from Pickup Day)', function() {
		browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')

	browser.driver.findElement(protractor.By.xpath('//select[contains(@name,
		"pickup_day")]')).sendKeys('30');
		browser.driver.findElement(protractor.By.xpath('//select[contains(@name
		, "pickup_month")]')).sendKeys('Dec 2014');
		browser.driver.findElement(protractor.By.xpath('//select[contains(@name
		, "dropoff_day")]')).sendKeys('17');
		browser.driver.findElement(protractor.By.xpath('//select[contains(@name
		, "dropoff_month")]')).sendKeys('Dec 2014');
		browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
		"search_btn")]')).click();
		util.retrieveText(protractor.By.xpath('//p[contains(@id,
		"result")]')).then(function(response) {
			if (response != undefined && response.length > 0) {
				expect(response).toEqual('0 cars found. Page 1, listing
				results 1- 0.');
			} else {
			util.retrieveTextForInput(protractor.By.xpath(
```

```
                '//p[contains(@id, "result")]')).then(function(response) {
                        expect(response).toEqual('0 cars found. Page 1,
                        listing results 1- 0.');
                });
        }
    });
});

it('TC-3: Successful Registration', function() {
    browser.driver.findElement(protractor.By.xpath('//a[contains(@id,
    "register")]')).click();
    browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
    "username_")]')).sendKeys('fatih');
    util.retrieveText(protractor.By.xpath('//p[contains(@id,
    "username_val")]')).then(function(response) {
        if (response != undefined && response.length > 0) {
            expect(response).toEqual('Username is not available');
        } else {
            util.retrieveTextForInput(protractor.By.xpath(
            '//p[contains(@id, "username_val")]')).then(function(
            response) {
                expect(response).toEqual('Username is not
                available');
            });
        }
    });

    browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
    "username_")]')).clear();
    browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
    "username_")]')).sendKeys('e190424');
    browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
    "password_")]')).sendKeys('123456');
    browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
    "confirm_pass")]')).sendKeys('123456');
    browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
    "fullname")]')).sendKeys('Fatih Isler');
    browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
    "email")]')).sendKeys('s.fatih.isler');
    util.retrieveText(protractor.By.xpath('//p[contains(@id,
    "email_val")]')).then(function(response) {
        if (response != undefined && response.length > 0) {
            expect(response).toEqual('Required');
        } else {
            util.retrieveTextForInput(protractor.By.xpath(
            '//p[contains(@id, "email_val")]')).then(function(
            response) {
                expect(response).toEqual('Required');
            });
        }
    });

    browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
    "email")]')).clear();
```

```
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "email")]')).sendKeys('s.fatih.isler@gmail.com');
        util.retrieveText(protractor.By.xpath('//p[contains(@id,
        "email_val")]')).then(function(response) {
                if (response != undefined && response.length > 0) {
                        expect(response).toEqual('ok');
                } else {
                        util.retrieveTextForInput(protractor.By.xpath(
                        '//p[contains(@id, "email_val")]')).then(function(
                        response) {
                                expect(response).toEqual('ok');
                        });
                }
        });

        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "occupation")]')).sendKeys('Software Engineer');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "gender_m")]')).click();
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "birth_day")]')).sendKeys('24');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "birth_month")]')).sendKeys('08');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "birth_year")]')).sendKeys('1986');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "register")]')).click();
});

it('TC-4: Reservation Without Login', function() {
        browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "search_btn")]')).click();
        util.retrieveText(protractor.By.xpath('//p[contains(@id,
        "result")]')).then(function(response) {
                if (response != undefined && response.length > 0) {
                        expect(response).toEqual('9 cars found. Page 1, listing
                        results 1- 9.');
                } else {
                        util.retrieveTextForInput(protractor.By.xpath(
                        '//p[contains(@id, "result")]')).then(function(
                        response) {
                                expect(response).toEqual('9 cars found. Page 1,
                                listing results 1- 9.');
                        });
                }
        });

        browser.driver.findElement(protractor.By.xpath('//*[contains(@id,
        "car_11")]')).click();
        util.checkVisibility(protractor.By.xpath('//p[contains(@id,
        "error")]')).then(function(response) {
                expect(response).toEqual(true);
        });
```

```
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "username")]')).sendKeys('e190424');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "password")]')).sendKeys('123456');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "login_btn")]')).click();
        util.checkVisibility(protractor.By.xpath('//p[contains(@id,
        "error")]')).then(function(response) {
                expect(response).toEqual(false);
        });
});

it('TC-5: Pickup and Dropoff Specification', function() {
        browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "pickup")]')).sendKeys('Ankara');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "search_btn")]')).click();
        util.retrieveText(protractor.By.xpath('//p[contains(@id,
        "result")]')).then(function(response) {
                if (response != undefined && response.length > 0) {
                        expect(response).toEqual('4 cars found. Page 1, listing
                        results 1- 4.');
                } else {
                        util.retrieveTextForInput(protractor.By.xpath(
                        '//p[contains(@id, "result")]')).then(function(
                        response) {
                                expect(response).toEqual('4 cars found. Page 1,
                                listing results 1- 4.');
                        });
                }
        });

        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "pickup")]')).sendKeys('Istanbul');
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "search_btn")]')).click();
        util.retrieveText(protractor.By.xpath('//p[contains(@id,
        "result")]')).then(function(response) {
                if (response != undefined && response.length > 0) {
                        expect(response).toEqual('5 cars found. Page 1, listing
                        results 1- 5.');
                } else {
                        util.retrieveTextForInput(protractor.By.xpath(
                        '//p[contains(@id, "result")]')).then(function(
                        response) {
                                expect(response).toEqual('5 cars found. Page 1,
                                listing results 1- 5.');
                        });
                }
        });

        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
```

```
		"pickup")]')).sendKeys('Izmir');
		browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
		"search_btn")]')).click();
		util.retrieveText(protractor.By.xpath('//p[contains(@id,
		"result")]')).then(function(response) {
			if (response != undefined && response.length > 0) {
				expect(response).toEqual('9 cars found. Page 1, listing
				results 1- 9.');
			} else {
				util.retrieveTextForInput(protractor.By.xpath(
				'//p[contains(@id, "result")]')).then(function(
				response) {
					expect(response).toEqual('9 cars found. Page 1,
					listing results 1- 9.');
				});
			}
		});

		browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
		"pickup")]')).sendKeys('Ankara');
		browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
		"dropoff")]')).sendKeys('Istanbul');
		browser.driver.findElement(protractor.By.xpath('//select[contains(@name
		, "pickup_hour")]')).sendKeys('08:00');
		browser.driver.findElement(protractor.By.xpath('//select[contains(@name
		, "dropoff_hour")]')).sendKeys('08:00');
		browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
		"search_btn")]')).click();
		util.retrieveText(protractor.By.xpath('//p[contains(@id,
		"result")]')).then(function(response) {
			if (response != undefined && response.length > 0) {
				expect(response).toEqual('4 cars found. Page 1, listing
				results 1- 4.');
			} else {
				util.retrieveTextForInput(protractor.By.xpath(
				'//p[contains(@id, "result")]')).then(function(
				response) {
					expect(response).toEqual('4 cars found. Page 1,
					listing results 1- 4.');
				});
			}
		});
});

it('TC-6: Sucessful Reservation', function() {
		browser.driver.get('http://localhost:7080/CarRentalSystem/reserve.jsp')
		browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
		"search_btn")]')).click();
		util.retrieveText(protractor.By.xpath('//*[contains(@id,
		"car_01_info")]')).then(function(response) {
			if (response != undefined && response.length > 0) {
				expect(response).toEqual('BMW 3 Series (2006)');
			} else {
				util.retrieveTextForInput(protractor.By.xpath(
```

```
                        '//*[contains(@id, "car_01_info")]')).then(function(
                        response) {
                                expect(response).toEqual('BMW 3 Series (2006)');
                        });
                }
        });

        browser.driver.findElement(protractor.By.xpath('//*[contains(@id,
        "car_01_info")]')).click();
        browser.driver.findElement(protractor.By.xpath('//input[contains(@id,
        "confirm")]')).click();
        browser.driver.findElement(protractor.By.xpath('//a[contains(@id,
        "reservations")]')).click();
        util.checkVisibility(protractor.By.xpath('//p[contains(@id,
        "status")]')).then(function(response) {
                expect(response).toEqual(true);
        });

        util.retrieveText(protractor.By.xpath('//p[contains(@id,
        "status")]')).then(function(response) {
                if (response != undefined && response.length > 0) {
                        expect(response).toEqual('Active');
                } else {
                        util.retrieveTextForInput(protractor.By.xpath(
                        '//p[contains(@id, "status")]')).then(function(response)
                {
                                expect(response).toEqual('Active');
                        });
                }
        });
    });
});
```

APPENDIX D: USER SURVEY FORM

Please fill given survey after using ATCGES-WEB and ATCGES-WEB-XPath Finder tools. In order to get detailed information about each survey category, you can visit http://en.wikipedia.org/wiki/Heuristic_evaluation page or search "Nielsen's Heuristic" at google.

Visibility of System Status: System should aware of user about system status by giving proper information in reasonable time

Grade(1-5)	Comment

User Control and Freedom: The easiness of exiting an undesired or mistakenly reached state in system. (Undo, Redo)

Grade(1-5)	Comment

Error Prevention: Inform user about process to be executed by showing confirmation boxes or similar components before commit

Grade(1-5)	Comment

Aesthetic and Minimalist Design: Dialogues and visual components do not give any irrelevant information

Grade(1-5)	Comment

Overall Usability and Difficulty:

Grade(1-5)	Comment

APPENDIX E: PROTRACTOR CONFIGURATION FILE

```
exports.config = {

    // The address of a running selenium server.
    seleniumAddress: 'http://localhost:4444/wd/hub',

    // Browser Driver Settings
    chromeOnly: true,
    chromeDriver: 'chromedriver.exe',
    capabilities: { 'browserName': 'chrome'},

    // The name of test cases to be executed and
    // their relative paths
    suites: {
        full_test: '../full_test.js'
    },

    // Test Framework Settings
    framework: 'jasmine',

    jasmineNodeOpts: {
        onComplete: null,
        isVerbose: true,
        showColors: true,
        includeStackTrace: false,
        defaultTimeoutInterval: 100000
    }
}
```

TEZ FOTOKOPİSİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü ☐

Sosyal Bilimler Enstitüsü ☐

Uygulamalı Matematik Enstitüsü ☐

Enformatik Enstitüsü ☑

Deniz Bilimleri Enstitüsü ☐

YAZARIN

Soyadı : İŞLER
Adı : Süleyman Fatih
Bölümü : Bilişim Sistemleri

TEZİN ADI (İngilizce) : **Automated Test Code Generation and Execution System for Web (ATCGES-WEB)**

TEZİN TÜRÜ : Yüksek Lisans ☑ Doktora ☐

1. Tezimin tamamından kaynak gösterilmek şartıyla fotokopi alınabilir. ☐
2. Tezimin içindekiler sayfası, özet, indeks sayfalarından ve/veya bir bölümünden kaynak gösterilmek şartıyla fotokopi alınabilir. ☑
3. Tezimden bir (1) yıl süreyle fotokopi alınamaz. ☐

TEZİN KÜTÜPHANEYE TESLİM TARİHİ : …….