Automation Testing Framework for Web Applications with Selenium WebDriver: Opportunities and Threats

Elior Vila
University of Elbasan "A. Xhuvani"
Faculty of Natural Sciences
Rinia Str., 3000 Elbasan, Albania
+355692820329
elior.vila@uniel.edu.al

Galia Novakova
Sofia University, Faculty of
Mathematics and Informatics
5 James Boutchier Str., 1164
Sofia,Bulgaria
+359885043978
galianovak@gmail.com

Diana Todorova
Sofia University, Faculty of
Mathematics and Informatics
5 James Boutchier Str., 1164
Sofia, Bulgaria
diana.a.todorova@gmail.com

ABSTRACT
The present paper discusses the need of automation testing in the process of software development, in order to provide high quality, robust and reliable software product. It answers the question why automation testing plays such a significant role in software development lifecycle as well as why not to use already existing automation testing tools when testing web applications and why it is better to create automation testing framework. Some reliable approaches how to build a testing framework are investigated. Selenium WebDriver tool is pointed out as appropriate solution when creating such framework and its wide use is outlined. Moreover, the paper provides analysis and detailed list of opportunities and threats of using Selenium WebDriver tool. The paper concludes by providing arguments for the value of the creation of automation framework for Web applications with Selenium WebDriver.

CCS Concepts
• Software and its engineering → Software testing and debugging

Keywords
Automation testing tool; software product; testing framework; web application; Selenium WebDriver; automation testing script.

1. INTRODUCTION
The aim of the this paper is to explain in details why automation testing is essential and what opportunities give the creation and usage of automation testing framework with Selenium WebDriver. Some of the best practices to be followed are suggested when selecting the tools and developing the automation framework so as to maximize the efficiency of developed product.

Nowadays the range of available web applications is incredible. The demand and supply of software products have increased. As the quality of software products matters significantly, software quality has become an inevitable part of software development lifecycle. In-built software quality helps meeting the specified requirements and the budget and deadlines.

As a result of dramatically improving development productivity, the pressure on testers to perform faster has increased. In addition, agile methods "are fast becoming the adopted development methodologies commercially" and project requirements are changing constantly [1]. Testers are asked to test more functionalities of the application under different conditions in lesser time. As different versions of software are released, the new features need to be tested manually, and old versions also need to be tested under regression testing [2]. If the same tests have to be executed many times, automation is a better option than doing it manually.

Automation software testing uses tools to run tests based on algorithms to compare the program actual outcomes with the requirements' expected outcomes. The usage of tools facilitates the work progress, because tools perform faster than human beings. Simple repetitive test cases may be given to a tool which will execute them much faster than a person can do it. If continuous integration system is used, tests can be exercised 24/7. Moreover there are some test scenarios that could not be reproduced by the tester manually. In this case an automation testing tool is needed. For example, when concurrent/parallel testing should be simulated a tool has to be used. A tester could perform only sequential test runs.

Variety of test automation tools is considerable. Selenium is probably the most widely-used open source solution for testing web applications [3]. One of the opportunities that Selenium WebDriver gives is the possibility to create custom framework for automation testing of web applications. Creation of automation testing framework with Selenium WebDriver could reduce time for development, increase the return on investments (ROI) and minimize the risk. Sometimes using COTS (commercial-off-the-shelf) automation testing tools can lead to some failure points in time:

- Recorded brittle tests - using recording tools to record tests. Such types of tools are used because tests are created fast and easily. When something in the system is changed, tests should be re-recorded again or should be edited manually in order to stop failing. In time people who update the tests give up because it needs more and more time to keep them up to date. In such situation using a recording tool does not worth the cost.

ICAIP 2017, August 25–27, 2017, Bangkok, Thailand

ACM ISBN 978-1-4503-5295-6/17/08…$15.00
DOI: https://doi.org/10.1145/3133264.3133300

- Writing tests like code - when automation tests are in program language they look as part of the application. This ends up to complicating the tests. As a result only developers and small number of QAs could update and expand the list of test cases because others do not even understand what the tests are verifying. It becomes tough to maintain them and as a result people stop using them.
- Creating tests that are not building a framework - creation of tests starts good but they fall to the same problem. When there are some UI or functional changes in the application, multiple tests that were dependent should be fixed. Creating a framework can solve and prevent such problems.

2. WEB TESTING

The high interest and popularity of the Internet have produced significant increase of Web applications' demand. Users are optimistic about the applications which are required to be more stable, reliable, secure, browse compatible and many other requirements.

Testing that is focused entirely on testing of a Web application is called Web testing and it is the only solution to attain and insure high level of quality of an application. Because of the rapid development of applications, the less time for testing and frequent releases, it is impossible to have exhaustive testing of an application. Developers usually disregard the need of thorough testing because it is thought to be time consuming and not repayable. This manner has negative influence over testing process and quality of the product. That is way more efficient and economic effective test approaches of verification and validation of quality in short terms become necessary [4].

Quality assurance is core activity in software development process and it is part of each phase of the development life cycle. Testing is not just a single activity, but it is a process. Steps that present testing life cycle correspond to fundamental steps of the development life cycle. Testing starts from test planning, analysis and design of test cases followed by implementation and execution of the test cases. Next are evaluation of test outcomes and preparation of final test report.

Quality assurance defines activities that prevent and fix defects in the time of software development. Testing defines many testing strategies that help to ensure quality of the product under development. Fundamental quality assurance activities include taking view of the requirements, auditing the results of control measurements and analyze quality performance. Quality of the software product is defined by validating the coincidence of user requirements and expectations, and implemented functionalities. The most important thing of a software system is to satisfy the customer and the user requirements.

Testing process is integral part of the software development life cycle. Testing process includes some fundamentals as defining the scope of testing, test strategy, quality assurance engineers' responsibilities and activities. Testing is performed on few testing layers, presenting the test cycles:

- Unit testing – Testing at a unit level. Includes testing of methods and classes. It is usually done by developers because testing is performed on code level. It tests low-level components to verify that they are doing what developers are expecting.
- Module testing – Performed for large and complicated systems that have few modules. Otherwise, it overlaps with Unit testing. A module is fully formed of chunks of coherent code. Testing in modules can be automated and it facilitates the debugging as well.
- Integration testing – Performed after integrating system modules or when two or more systems that interact with each other. It still focuses on the code level. Tests are written in the code and directly manipulating it. Code is not tested through the user interface. No fake data, stubs or drivers are needed because classes and modules that are interacting are already implemented. Value of integration testing is that classes and modules are checked whether working as expected in early phase of development and if some issues are found, the cost of fixing them is lower.
- System testing – Behavior of the whole system is tested. System is tested from the perspective of a user. System testing verifies that the system to be delivered meets its purpose, user requirements and needs. This testing layer includes functional and non-functional testing. Time for regression testing is saved as well.
- Acceptance testing – On this testing layer is verified how the system performs in production environment, whether it works as expected or not. The end users of the system execute testing.

All types of testing are complementary, not exclusive. Unit, Module and Integration testing fall into the round of the white-box testing because the person who is testing sees the code. Whereas System and Acceptance testing are part of the black-box testing approach because the tester is interacting with the code at the same level as the user is going to use the application. The system under test is treated as a black box.

When iterative development of the software is followed, need of testing increases significantly. After the end of iteration, application should be verified that changes have been applied successfully. It is needed to verify also that the old functionalities, which have been implemented in the previous cycles of development, are still working as expected and no regression has been introduced. That is why each development cycle ends with regression testing. Regression testing is executed after implementation of new functionality, after implementation of change requests or refactoring. In summary after any change of the program, regression testing is required. The purpose of it is to catch unexpected consequences, bugs that might have been introduced into a new build or delivery. The objective is to confirm that components that used to be working are not failing and to ensure that previously fixed defects have not arisen. In such situations automation testing is the best solution.

Automation testing uses specialized software to control the execution of test scripts and to compare the actual outcomes with predicted outcomes. The main advantages of automation testing are:

- Efficiency - Time for execution of automated test scripts is less than executing them manually because they are run by tools that facilitate testing.
- Repeatability - The same list of test cases can be executed again in absolutely identical way as the previous time. Using tools for test automation eliminates the risk of human mistake when re-creating a problem in an application.

- Scope of testing - It becomes very easy to track what amount of code has been covered by the test cases and whether there are major functionalities that have not been tested. This could help to see if more tests should be created.
- Decrease of expenses - Using automation testing in the daily work might lead to lower down the expenses that otherwise are needed for manual testing.

Automation testing does not exclude the human agent and the necessity of manual testing because not every test case could be automated. There are functionalities that do not allow automation. Also, not every type of project is appropriate to be automated.

A. Automation Testing Methodology

Automation testing framework is defined as multitude of suppositions, conceptions and practices that ensure the support for automation testing [5]. When a methodology for automation testing is adopted, there are few things that should be taken into consideration [6]:

- Testing framework is external and independent from the application.
- Testing framework is scalable and maintainable
- More than one test script could be run in parallel without any conflicts.
- Testing framework ensures additional level of abstraction for work with testing environment.

Recording and execution of test scripts

The methodology "Record and Playback" is based on automatic generation of executable script that describes the user flow activities when interacting with the application under test. In a later moment, when created test scripts are executed, the steps that have been recorded are repeated in the same way and order, as when the script was created. That's way this methodology of automation testing is called "Record and Playback". This method for automation testing is thought to be the most cases not efficient. Some of its characteristics are:

- Expensive tools
- Hard coded data in the scripts - Test scripts contain hard coded input and output data. When test scripts are created with record-playback tool, the data that is entered in the fields is captured and the output results are saved. Every next time when a script is executed, the same input data is entered in the fields and the same output results.
- Difficult & expensive maintenance - after each change of the system under test, test scripts should be re-recorded or fixed manually in the code, otherwise they will fail every time when test scripts are run. Time is invested into maintenance instead of creation of new test scripts.
- The process of recording is hard - if unexpected message appears while recording the test script, the whole test should be recorded again from beginning.
- All these leads to the case when recorded test scripts are run only few times in the way they were created. After that extra time should be spent in reworking and maintenance.
- Parameterization of input and output data

When methodology of parametrization of input and output data named Data-driven methodology is used, scripts are created and developed in the same language as the automation testing tool that is used. After that scripts are updated to be working with variable input and output data. Dates might be stored in a file or into database. After that when tests are executed, data is read from the repository. Characteristics of the methodology:

- Less automation testing scripts are necessary
- This methodology has the same disadvantages as the previous (above) methodology – in summary, hard coded data, difficult and expensive maintenance.
- Functional Decomposition

When Functional Decomposition method is applied, hierarchical structure and module design are used. Functional Decomposition is appropriate for complex business processes that need to be simplified. Each test is broken down into parts that are developed as independent scripts. By dividing a flow into steps, it becomes simplified and the work is facilitated. In that way tests are generated as sequence of scripts that can be included in other testing scenarios. The characteristics of Functional Decomposition method are:

- Time and effort for creation of automation tests is decreased by reusability of the test scripts.
- Effective maintenance of the scripts – when something in the system has changed and a change in the scripts is required, only the scripts that are affected should be updated.
- Extensive rework of Process Modeling methods
- Complex interfaces – Complex and numerous interfaces could be created, which might lead to confusion.
- Internally focused
- Some important component could be missed

The idea of listing and reviewing in the details the above methods for automation testing is to identify and show their disadvantages and to point the necessity to create a framework for automation testing of Web applications. "A growing trend in software development is to use testing frameworks such as the xUnit frameworks which allow the code to conduct unit tests to determine whether the various sections of the code are acting as expected under given circumstances" [1].

B. Telerik Testing Framework

The proposed framework for automation testing is not unique and the only one available solution. Many enterprise organizations develop such frameworks as internal project, in order to facilitate the testing process. There are some tools and frameworks for automation testing accessible on the market. For example, Telerik Company has created a framework for automation testing that is very similar to another framework Selenium WebDriver [7]. However, one of the greatest disadvantages of the Telerik framework is that it is costly. The user has to pay a fee which starts from $169/month although there is a 30-day trial version.

Selenium and Telerik Testing Frameworks are only APIs. In order to create a test suite, a custom framework should be developed on the top of selected framework. They just allow the user to send actions to the browser.

Many of the testing automation tools, different from Selenium, are harder and more expensive to maintain than Selenium framework is, some of them use hard coded data or automation of tests is complex. While developing the Selenium framework, approaches that are followed should satisfy the KIS rule. In other words, the "keep it simple" rule - develop separated framework, create simple tests and let the tests to drive the creation [8].

3. AUTOMATION TESTING FRAMEWORK

The framework architecture could be designed to provide the ability to create automation tests for different web applications. The programming language used for applications' development could not be taken into consideration when selecting a web application that is going to be automated, because Selenium performs in-browser testing. Automation tests are going to be written in a special way, simplified and comprehensible for a large group of people. New classes and methods could be created to expand and upgrade the functionality of existing ones. They could be designed in a way to be reusable when creating automation tests to verify the functionality and quality of different web applications [9][10].

There are few essential approaches that are better to be followed when creating a framework for automation testing. The most important and the most valuable things that should be done in order to reach a success are as follows:

- Create a separate automation framework - Having a separate automation framework is one of the best practices in automation testing. Tests that are created should not directly work against the web application. It is not a good practice because if something has changed in the web application, all tests are going to fail and each test has to be updated.
- An appropriate approach, when building an automation framework is to seat between the web applications and the tests, i.e. it should be some kind of a proxy. In that way when a test wants to do something it is going to call a specific method in the framework. The framework on its own is going to know how to process the request in the web application. If a framework is used to control the tests and something has been changed in the web application, this will cause only the framework to be broken. When it is updated, all of the tests are going to work properly again. The key thing to have a successful automation solution is to have separate testing framework.
- Create simple tests – Tests should be implemented in a simple and understandable way. Simple terms and simple language have to be used. One of the key goals that we seek to achieve while creating the framework is to make it as easy as possible. The reason why tests should be built in a simple way is that if we have complex tests they are going to be hard to maintain and less people would be able to update and upgrade them. Others would not understand them, so tests will become obsoleted and broken. If this happens framework will be in heaviness instead of increasing the efficiency. If tests are kept simple everyone will be able to understand what the tests are doing, which functionality they are verifying and what their purpose is. More people are going to be able to expand the list of automated tests by automating more test cases. If only the creator knows how to use the framework, work could not be distributed among the QA team.
- Let the tests drive the creation of the framework – if the framework is created first a lot of time is spent without having any tests and results that could be shown. Another problem is that lots of guesses should be made about how the framework is going to be used and what tests will be created. There is high possibility that some piece of the framework will be missed or wrong. That is why an initial design how the framework will work and interact with web application is needed. In that way the main thing that will drive the creation of the framework will be the tests themselves. First, the tests are going to be written into the framework and after that the code that is going to make the tests to work. This approach corresponds to the essence of Test Driven Development-TDD. In that way tests will be simple, expressive and maintainable. Framework is required to work with the test, not vice versa.

C. Framework architecture

Implemented software framework for automation testing of Web applications follows two-tier architecture. The two-tier architecture has two layers:

- Client layer – It contains all the functional automated tests, created and run by the users of the framework.
- Business layer – It contains all the business logic of the Web application under test, its pages or main functionalities. It also contains methods to define the core things like step definition and verification, platform and browser selection, report generation, etc.

The project as presented in figure one, is divided in few packages in order to achieve abstraction, modifiability and high maintainability. Available packages are:

- com.webdriver.core – this package holds the core functionality, used to step on and upgrade the framework. Fundamental functionalities that could be found and edited by the user of the framework are definition of the platform, the browser that will run the tests, timeout period, parallel or sequential execution of the test scripts and others.
- com.webdriver.dataprovider – in this package is defined the functionality about reading from an excel file. The ability to load user credentials automatically from the data provider is implemented in the classes.
- com.demoqa.pages – this package contains identification of the pages of web application. Pages fit the exact pages or just present logical division of main functionality. Different element locators of Web pages are used to locate the objects on the pages of the Web application under test. Methods that are implemented in the package perform user actions. Web elements that have been located are used to execute all actions in the created methods.
- com.demoqa.tests – Package contains only completed and prepared tests for execution. In the tests are included the methods and functions from the "com.demoqa.pages" package. Methods that indicate the start and the end of a test are called as well as the methods for start and end step of a test case.

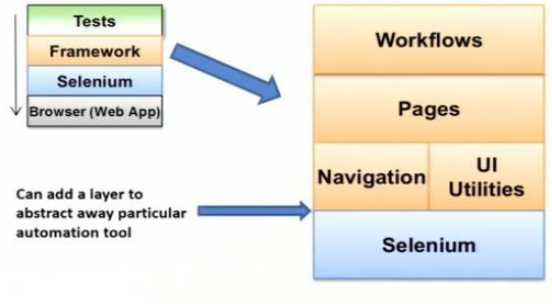

Figure 1. Building blocks of the framework

In order to build maintainable framework and programming code, Page Object Model (POM) pattern is adopted to develop Selenium WebDriver tests. "This pattern helps in enhancing the tests, making them highly maintainable, reducing the code duplication, building a layer of abstraction, and hiding the inner implementation from tests".

Page Object Model is a design pattern that refers to the idea of mapping the model of the testing framework to the model of the pages of the application under test. Web pages are forming the structure of the framework by creating the corresponding classes. A class has the same functionalities that the page has. Sometimes a class might correspond to a section or main functionality of the Web application instead of to a Web page. Using the both approaches, pages and tests are mapped. The class is responsible for getting all elements on the Web page. After that, it manipulates with them when creating the high-level methods called in tests. Page Object Model is design as abstract API. Framework for testing is designed in a way as the user is going to interact with the Web page. It is mapped to logical objects. By doing that way, writing of test cases becomes easier [11].

Page Factory is also a pattern that extends POM. It makes usage of Page Objects easier and simpler. Applying the Page Factory pattern gives clear vision of the objects residing on the page and operations that could be performed.

In order to produce the contents and the relation between different packages in the framework, an example with one of the implemented functionalities is presented in the paper. The functionality that is selected is frequently met in Web applications and the type of data entered is very similar – it is about registration form. The registration form that is used for trial is in http://demoqa.com/ site. The purpose of this application is to practice automation testing skills. It contains different forms, buttons and fields, which make it appropriate environment for learning and testing.

"RegistrationPage" class as presented in figure two is placed in the first tier of the 2-tier architecture – in "com.demoqa.pages" package. Selenium WebDriver provides advanced techniques for locating elements on Web pages. One of the good practices in software development of Web applications is to add attributes as ID, Name and Class to all Web elements. That makes the Web application testable and it meets the accessibility standards. Unfortunately, not all Web applications follow these standards. Sometimes only parts of an application follow the standards, but not the entire one. Then it is needed to use advanced locator strategies like CSS selector and XPath. Comparing both CSS selector and XPath, CSS selector is faster and performs better than XPath.

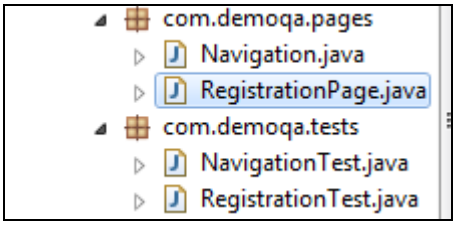

Figure 2. RegistrationPage class into com.demoqa.pages package

Three locator strategies are used in the applied example – ID, Class and XPath. "@FindBy" annotation, provided by Selenium, is used with specified type of locator strategy.

User interface

Current version of the framework for automation testing of Web applications does not have user interface (UI) design. It is just available as Java project, which has a package with automated tests.

Control and monitoring of tests running could be performed in Eclipse IDE. If a user wants to run a class with tests or a suite of tests, he/she has to select the suite or the class, write click with the mouse on it and select Run As> TestNG Suite or Run As> TestNG Test. Then suite or class with tests is run. In the Console of Eclipse IDE are shown the results from tests execution. In Console is displayed status of the tests - how many tests have Passed, Failed or Skipped. If any problem has appeared, Stack Trace is printed.

Additional modules

Few additional modules are needed for the implementation of the framework, such as:

- TestNG framework - TestNG is required and it is installed as a plug-in in Eclipse IDE.
- XML editor – In order to view and edit excel files, which are sources of test data, XMLEspresso XML editor Eclipse plug-in is installed.
- Firebug – Plug-in is installed in Firefox browser to inspect elements on the Web pages.
- FirePath – It is Firefox plug-in, which is used to take the absolute path of an element easier.

Defining the right architecture in the beginning, before starting the implementation of the framework, gives the ability to modify, maintain and scale it in a later moment without redundant effort.
N-tier architecture and split functionalities in logical modules help to have a clear vision about the structure of the framework. Replacement and upgrade of pieces of code becomes easier as well.

Defining the appropriate framework architecture and structure according to the needs of the project is very important part of project realization, because it could be really helpful and productive at the end.
Creation of automation testing framework for Web applications with Selenium WebDriver has some advantages and threats [12][13]. The opportunities are as follows:

- Selenium is open source tool - It is free software and it is highly flexible and there are many ways to add functionality. Source code is available for those who want to download and modify it. It supports active community that resolves issues and queries related to Selenium [14].

- Web browser support - Selenium works not only with Internet Explorer as many other tools do, but it also supports major browsers as Firefox, Google Chrome, Safari, Opera, HtmlUnit and the mobile - Android and iOS.

- Platform support - Selenium supports Windows, Max OS X and Linux. The cross-platforms feature makes easier testing of web applications in various environments.

- Programming language support - Selenium supports few OOP (object-oriented programming) languages including Java, C#, Ruby, Python, PHP. Quality assurance engineers have the opportunity to choose the programming language that they are going to adopt to write test scripts on.

- Support for IDEs and Testing Framework - Selenium allows choosing between few IDEs for development of test scripts.

These are Eclipse, IntelliJ IDEA, NetBeans and Visual Studio.

- Simulate a real user working with a browser - WebDriver uses real browser to access the web application which do not differ from activities of an ordinary real user. When web pages are loaded, browser loads all web app resources, executes all the java scripts on the page, all cookies are created and etc. That is why it is hard to identify whether it is a real user or a bot.
- Web pages with dynamic content are not a problem - Pages that generate dynamically their content can be still automated by using the appropriate locators.
- It is possible to create complete test automation suite.
- WebDriver is able to take screenshots.
- High maintainability of created automation test scripts – When the application under test has been changed and some or all of the automated tests start to fail; it is enough just to update the framework for automation testing and the tests will be fixed.

As any software tool, creating testing framework using Selenium WebDriver has some threats, as follows:

- Higher technical competence in a programming language is needed - If someone wants to build, update or upgrade an automation testing framework that is built on Selenium WebDriver, she/he has to possess good programming skills. Otherwise it would take much time for him/her to investigate how to implement something or he/she might bring in an issue.
- Could not create automation test cases immediately and from scratch - Some time for framework development is needed initially before creating the automation test cases. So there is a period of time from start using the Selenium tool to creating and running the automation test cases.
- Need to connect and work with third party frameworks and add-ons as TestNG, FirePath, Firebug and etc.
- Selenium WebDriver is time and resource consuming - When an automation test case is run, the web application needs to be started in the browser. This takes time and system memory because the whole web page should be loaded. Sometimes security issues might be caused.
- Bigger network traffic is generated - When web application is loaded in the browser, a lot of supplementary files are loaded (i.e. images, css, js files and so on).
- Program becomes quite large - program is linked with all Selenium WebDriver libraries and the driver executable needs to be installed for each browser.

D. Analyses of Test Results

When test run completes, a test report as HTML file is generated. It contains information about number of tests that have been run, their running time, chronological execution and whether some methods have been ignored. There is Reporter output section, which keeps information about Browser used for running the tests and its version. Also, each step of run tests is printed and it shows whether it has passed or failed. If a test has failed, problem that caused the failure is printed, step on which the test has failed is marked and a screenshot is taken. Results section contains summarized information about passed and failed test cases.

E. Experimental integration

When some functionalities of a Web application should be automated, already created framework could be used. The same project is used for a basis. New page package should be created, called com.<application_name>.pages. It will contain all pages of the Web application that is going to be tested. Another package for tests should be created as well. Its name should be com.<application_name>.tests. In that way, it is clear even from packages' name, which one is the automated application.

4. CONCLUSION

While comparing the opportunities and threats of Selenium WebDriver the advantages prevail over disadvantages. Selenium remains the most widely-used and preferred tool for creating automation testing framework. Selenium WebDriver offers a lot of possibilities for creating functional tests. It is flexible, extensible and can simulate a real user behavior. Created automation testing framework could be used in different platforms, browsers and gives the opportunity to create test scripts in diverse programing languages. Selenium features and opportunities become essential for the business because it could help meeting the needs, budget and test progress.

Web application has been selected as appropriate because it has typical and frequently used actions. As a result of surveys, this application has been chosen because of the fact that it has been created for test purposes and automation testing practices in particular. Research showed that it is not a good practice to use regular commercial Web applications for training as well.

Analyzing the main application's functionalities includes exploring, analyzing the most frequently used and the most important actions in the application to implement first. Registration form was selected for demonstration as most commonly met feature in different Web applications. Registration form contains the following actions - enter text in a text field, mark an option as selected, select a value from drop down list, click on a button, etc.

Building the framework for automation testing with Selenium WebDriver includes many subtasks related to implementation of classes and methods that are used to automate common functionalities. In this way, the automation testing framework could be easily applied to different Web applications and could be updated according the needs.

The proposed framework for automation testing of Web applications with Selenium WebDriver gives the opportunity to automate the manual processes that are already in place in small and large companies. It gives to quality assurance engineers the ability to verify and validate Web applications' functionality in shorter terms. The framework follows some of the best practices in order to facilitate automation of tests. Its use saves time, increases the investments and gives the opportunity to modify and enlarge features and functionalities to be tested.

Current version of the framework covers the basic needs for testing a Web application and has great areas to expand. It could be enlarged and improved in order to cover more functionality, features and needs, as follows:

- Expand the framework to handle more common functionalities and activities;

- Implement GUI of the framework in order to be more user-friendly;
- Integrate automated tests with Continuous Integration (CI) tool.

5. REFERENCES

[1] Conboy, K., Coyle, Sh., Wang, X. 2010. People over Process: Key challenges in Agile Development, *IEEE Software, Volume: 28, Issue: 4, (July-Aug. 2011), 48-57.*

[2] Limaye M. G., Software Testing 2009, Tata McGraw-Hill Education.

[3] Unmesh, G. 2012. Selenium Testing Tools Cookbook, Packt Publishing.

[4] Sonmez, J,. 2013. "Our Approach", Introduction, Creating an Automated Testing Framework with Selenium C#, (Pluralsight 2013).

[5] Cervantes, A., 2009. Exploring the use of a test automation framework, *Aerospace conference, 2009 IEEE, 7-14 March 2009.* DOI**:** 10.1109/AERO.2009.4839695.

[6] Sonmez, J,. 2013. "Why do it", Introduction, Creating an Automated Testing Framework with Selenium C#, (Pluralsight 2013).

[7] Stoil - Telerik Testing Framework vs Selenium, August 24, 2015 March 22, 2016 http://slashstars.com/telerik-testing-framework-vs- selenium/

[8] Sarkar, E. 2015. Is Selenimu a real threat to UTF?, http://ezinearticles.com/?Is-Selenium-a-Real-Threat-to-UFT?&id=9251731

[9] Pluralsight's videos "Creating an automation testing framework with Selenium".

[10] Introduction to Selenium WebDriver – Selenium Tutorial #8, 2017, http://www.softwaretestinghelp.com/selenium-webdriver-selenium-tutorial-8/

[11] Sonmez, J. 2012. Page Object Model Overview, Building a framework, Automated Web Testing with Selenium C#, (Pluralsight 2012).

[12] Shilov, M., 2013. Pros and Cons of using Selenium WebDriver for Website Scraping. http://scraping.pro/using-selenium-webdriver-for-website-scraping/

[13] Selenium documentation, http://docs.seleniumhq.org/docs/

[14] Sonmez, J,. 2013 "Types of Automated Testing", Introduction, Creating an Automated Testing Framework with Selenium C#, (Pluralsight 2013).