International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 11 (2018) pp. 9758-9767
© Research India Publications. http://www.ripublication.com

Automated Software Testing Framework for Web Applications

Milad Hanna[1], Amal Elsayed Aboutabl[2], Mostafa-Sami M. Mostafa[3]

[1]Teaching Assistant at Computer Science Department, Faculty of Computers and Information, Helwan University, Egypt.
[2]Associate Professor at Computer Science Department, Faculty of Computers and Information, Helwan University, Egypt.
[3]Professor of Computer Science, Faculty of Computers and Information, Helwan University, Egypt.

Abstract

Most of the time, customers request complex business logic to be implemented in software applications. Therefore, as long as business requirements grow, the pressure increases on the testing team to deliver the product with high quality in a very tight time. Manual testing is not suitable for critical and complex applications in terms of both human resources and time. Therefore, there is a strong need to propose an automated testing framework which could reduce the overall software testing time. Automation testing has been introduced to overcome manual testing problems. This study aims to propose a new automated testing framework for testing web applications that enhances the automating process. The proposed framework can save approximately about 75% of the total time/effort involved in the automation process using traditional automation and 21% compared to using Selenium IDE.

Keywords: Software Testing, Automated Software Testing, Test Data, Test Cases, Test Script, Manual Testing, Software Under Test.

INTRODUCTION

Automation of software testing is the process of creating a program (test script) that simulates the manual test case steps in whatever programming/scripting language [1] [2] with the help of other external automation helper tool [3] [4]. It is the process of automating the manual testing steps. Testing engineers have to implement and run a program to test the Software Under Test (SUT) [5]. In other words, it is developing toolkits to test the already implemented source code [6]. It aims at developing the testing phases to be automated [7]. Developing the application and test scripts are both development tasks, the first one is for the development of the application itself and the other is for developing the scripts that will be used to test the application.

Automating the execution phase of the software testing cycle is particularly the most popular approach in the automation field. Nowadays, not only automated software testing is important but also adding more toolkits to make testing phases fully automated by generation of test scripts [3] [8]. It increases the test execution speed as it can be used many times with the same effort. Developing test scripts take a long time to be implemented. However, after the test scripts are ready, the human tester can execute them automatically repeatedly on the SUT [9] [10]. Therefore, the benefit of automation will be gained on the long run. Yalla and Shanbhag [11] concludes that the usage of automated testing framework with automation tool can reduce the overall project testing effort.

A test automation framework is a set of predefined set of concepts, abstract ideas, assumptions, and implementations that provides support for software testers in automating software testing [12] [13]. It can also be defined as a set of encapsulated functionalities that facilitates the automation process itself [14]. These researches help a lot not only in controlling and monitoring the execution of business test scenarios [15] but also in increasing the reusability of the automated tests [16].

This work is motivated by previous studies [17] [18] [19] [20] where it is reported that there is a lack of studies which bridge the gap between theoretical and practical aspects in automated software testing. Petersen et al. [20] reported that about 45% of the testers are not satisfied with the current available automation frameworks in the market due to their poor features comparing to their project needs. In addition, they did a practical survey on both the benefits and limitations of automated testing to guarantee that their academic study is linked to the practical experience of software testers.

The objectives of the proposed Software Automated Testing Framework (SAT) framework are:

1. Enhancing and raising the collaboration between researchers and practitioners since automation frameworks help to transform automation from theory to practice.
2. Benefiting from both of record/playback scripting techniques (ease of initial development) and programmable scripting techniques (ease of maintenance).
3. Simplifying the test scripts maintenance process as the authors believe that even semi-automated testing frameworks will be still valuable and allow testing engineers to automate web applications testing in an easier manner.

Our proposed Software Automated Testing (SAT) framework overcomes the limitations of traditional automation techniques by extending the automation to involve all testing tasks. It translates the well formatted test case steps to reusable programmable test scripts, then the Selenium automation tool reads the auto generated test methods and runs them on the web browser. There is no need for any programming background to automate web applications since the source code generation step is the main responsibility of the framework. The tester will only add or update the generated testing steps. This eliminates the effort of creating automation projects from scratch. The main target of the proposed framework is to reduce the overall cost of the test automation process.

RELATED WORK

To build a successful automation project, the first step is to select an automation testing tool based on the application type to be tested and the license cost of the automation tool [21] [22]. Automation testing tools help testing engineers to easily automate the software testing phase. Commercial automation tools have higher cost than open source tools. However, this high cost will be worth the benefits for the tester since it provides the tester with full support which is not available in open source tools. On the other hand, the advantage of open source automation tools is that programmers always add continuous enhancements in these tools within their community free of charge. In addition, companies prefer to use open source tools because it is cost effective [23].

Selenium [24] and QTP [25] automation tools are the most commonly used tools in automated software testing. However, QTP is not always preferable because of its high license cost. On the other hand, since Selenium is an open source tool, it is more popular among testers. However, it is less user-friendly than QTP and requires a high level of programming knowledge. This Selenium limitation motivated us to develop our SAT framework which is built over it to be more user friendly. Traditional automation approaches are expensive and consume a lot of time in developing test scripts from scratch.

There are a lot of software testing automation tools which are available in the software market. These tools have similar core functions. However, they differ in functionality, usability and features. Table 1 summarizes various studies [21] [26] [23] [27] [28] [29] which compares between the most common automation tools:

Table 1. Comparison between Automated Testing Tools

Features	**Selenium**	**Quick Test Professional**	**Test Complete**
License Cost	Open source	Licensed. Very expensive, costs about 13000$	Licensed. Costs about 1999$
Application Type	Web applications only	Web and Mobile applications	Any type of application (web, desktop)
Record Playback	Support	Support	Support
Programming Language Support	Java .Net Perl PHP Python Ruby	VBScript JavaScript	VBScript JavaScript Delphi Script C++ C#
Platform Support	Windows MAC UNIX LINUX	Windows only	Windows only
Browser Support	All browsers	IE Firefox Chrome	All browsers
Technical Support	No official technical support	Good technical support	Good technical support
DialogBox Support	Partially supports	Supports all major kinds of dialog boxes	N/A
Creation Of Scripts	Not powerful as many actions are not recorded by the IDE	Powerful to some extent than selenium	N/A
Usage	Need experience and programming skills	Easy to learn/use/edit/ parameterize/playback the VB script	Need experience
Data-Driven Framework	Excel- CSV	Excel files Text files XML DB files	CSV Excel SQL
Report Generation	HTML	HTML	HTML, XML

PROPOSED SOFTWARE AUTOMATED TESTING FRAMEWORK

The proposed SAT framework can be represented as a new layer between testers and the existing automation tools. It is a desktop application that helps testers to automate the test script generation process [30] [31]. Automating this process will save a lot of testing effort and hence decrease the overall project cost [32]. The scope of the proposed SAT framework is restricted to automating the regression software testing phase for web applications only. Fig. 1 illustrates the proposed SAT Framework for web applications.

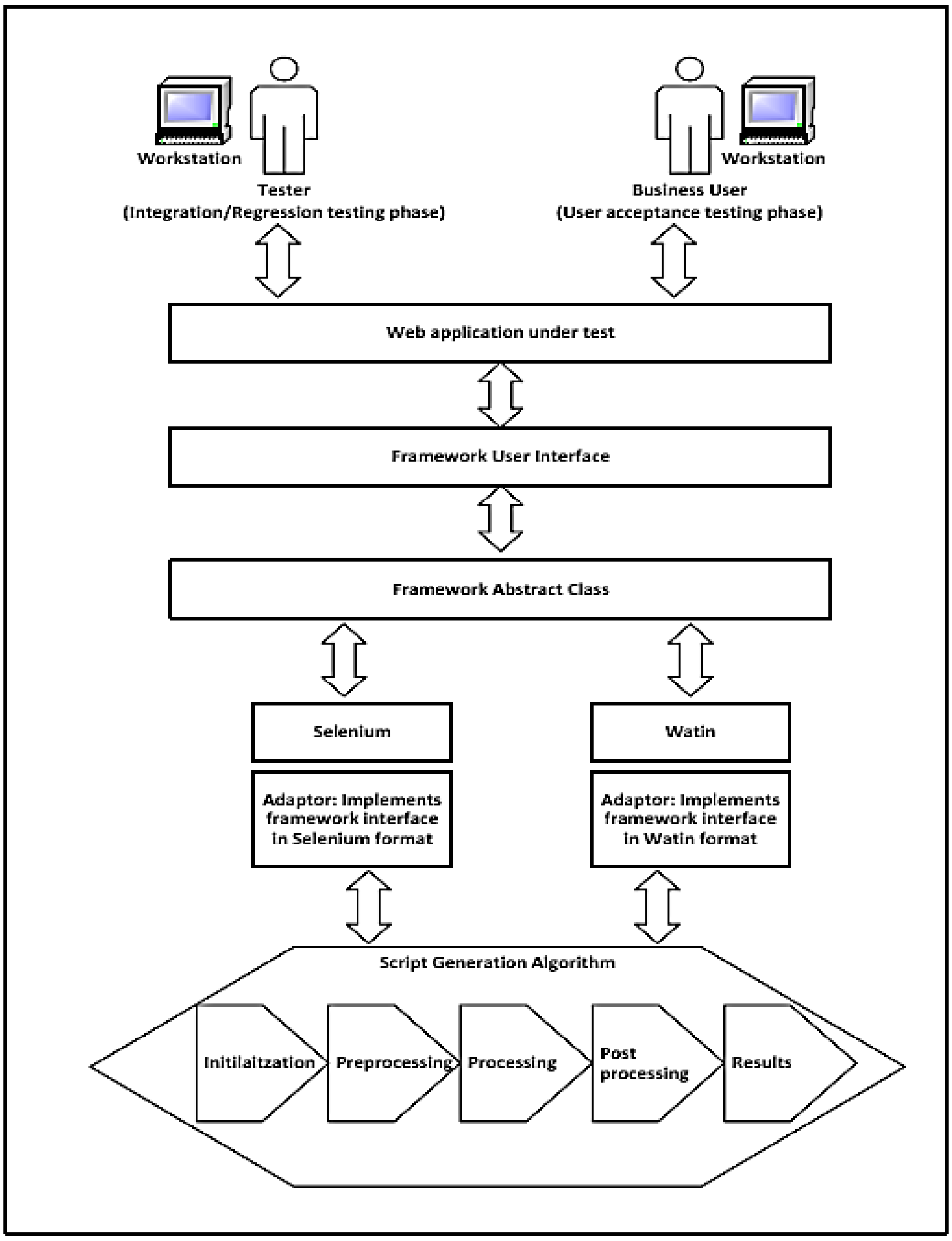
Workstation
Tester
(Integration/Regression testing phase)
Business User
(User acceptance testing phase)
Workstation
Web application under test
Framework User Interface
Framework Abstract Class
Selenium
Watin
Adaptor: Implements framework interface in Selenium format
Adaptor: Implements framework interface in Watin format
Script Generation Algorithm
Initialization
Preprocessing
Processing
Post processing
Results

Figure 1. Proposed Framework Model

The core framework engine has five main processing steps. The first step is the initialization step in which the SAT framework crawls the SUT web page to detect all the HTML controls in the web page that will be tested and make sure that each HTML control has a predefined list of values in the framework repository. If one control in the SUT does not have corresponding data values, then the framework asks the tester to create a list of proposed values that are matching with this new HTML control. These added values will be used later when the framework proposes the data value for this HTML control. In the preprocessing step, SAT framework selects one value for each HTML control that will be used during execution of test script. In the processing step, both Test Case Steps (TCS) sheet and test scripts are automatically generated based on both the

HTML controls in the SUT and the web automation tool. It is the core step in the framework engine. In the post-processing step, the tester should use the auto generated test script and run it over the SUT web pages. In this step, the framework will always validate that each input control in the test script still exists in the SUT. Finally, in the results step, results of running test scripts (that represent test cases) are listed to the tester or the business user. Our proposed SAT framework has a lot of features of some existing automation tools [21] [26] [23] [27] [28] [29]. It reduces the cost of the automation process by enhancing creation, reusability and maintainability of test scripts, which is the main objective behind test automation. Reducing test automation cost is important because manual testing cannot be executed repeatedly over the different testing phases. Therefore, there is a great need to convert these manual test cases into automated test scripts. Our proposed SAT framework automates the automation process by reducing the effort which is consumed in test scripts creation [23] [1]. Unlike the authors in [33] who proposed a test automation framework for avionics system, our proposed framework is independent of the SUT internal business details. Thus, the proposed framework can be used to automate any SUT without any additional cost.

The activity diagram in Fig. 2 illustrates the sequential activities of the proposed SAT framework. The tester should provide the framework with the URL of the SUT. Then, the framework opens this URL, and auto-generates the TCS sheet and the corresponding test script. The tester can update both files after their generation. Finally, the tester (or the business user) can run the auto-generated script on the SUT.

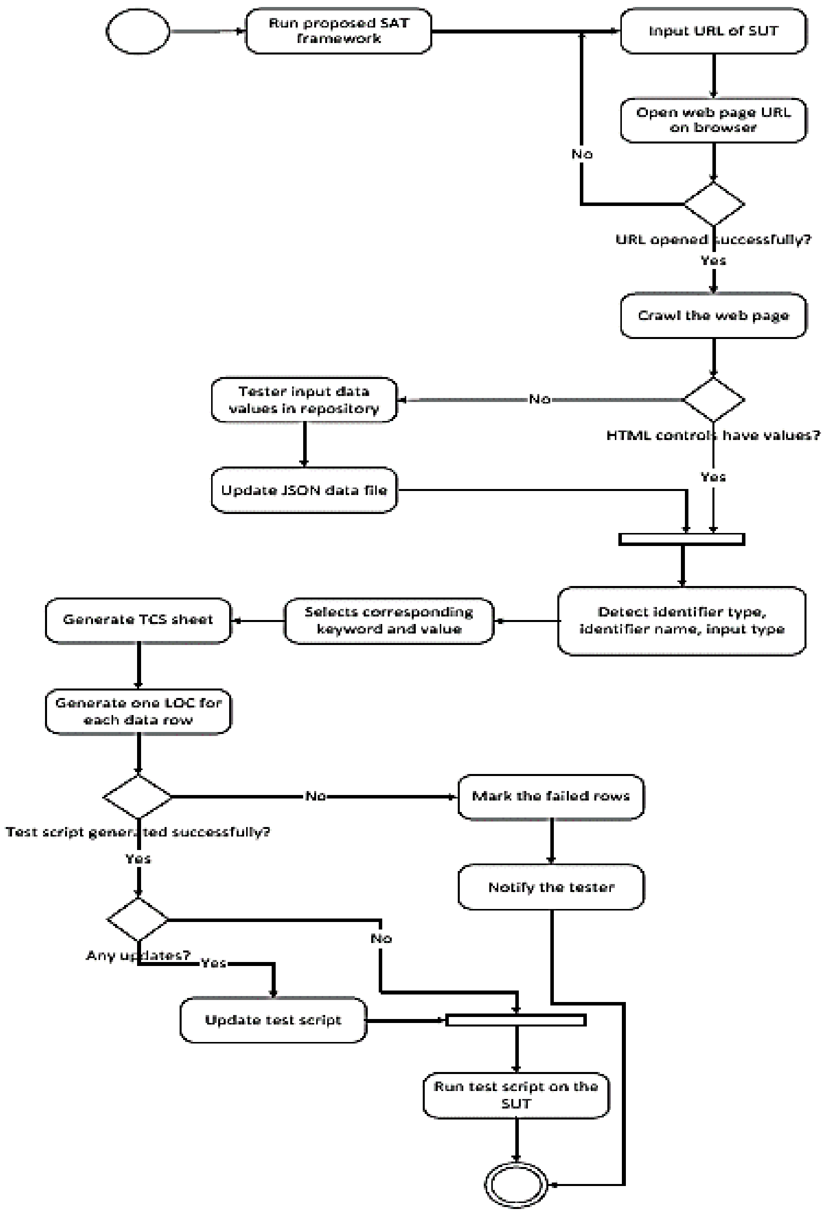
Run proposed SAT framework
Input URL of SUT
Open web page URL on browser
No
URL opened successfully?
Yes
Crawl the web page
Tester input data values in repository
No
HTML controls have values?
Yes
Update JSON data file
Generate TCS sheet
Selects corresponding keyword and value
Detect identifier type, identifier name, input type
Generate one LOC for each data row
No
Mark the failed rows
Test script generated successfully?
Yes
Notify the tester
No
Any updates?
Yes
Update test script
Run test script on the SUT

Figure 2. Proposed Framework Activity Diagram

PROPOSED FRAMEWORK MAIN FEATURES

Using any of the traditional scripting techniques (linear, data-driven, keyword-driven…etc.), the tester has to write down test case steps into detailed and complex sheet format. However, In the proposed framework, the algorithm can determine the appropriate keyword automatically according to the HTML input type. A keyword may be SetText / SelectValue / OpenURL / ClickButton /…….. etc. Therefore, this eliminates the effort of manually selecting the appropriate keyword that matches the HTML input control.

One of our contributions is improving the TCS sheet through transforming it into less detailed level. Hence, if a tester or business user chooses to use our SAT framework, the TCS file format will not require filling in any HTML control attributes manually. The SAT framework will auto-generate these technical details automatically for the tester. Simplifying the TCS sheet format in that way makes it easier for testers to use automation in the regression-testing phase and also for the business users to use it in the user acceptance-testing phase. Our proposed SAT framework engine is responsible for transforming these business attributes to their technical HTML object identifiers. These business attribute values are the labels which describe each HTML control in the SUT. By automating this step, testers will not need to worry about updating HTML control identifiers every time they update a field during maintenance activities for the software. Fig. 3 illustrates the transition from less to more automation scripting techniques.

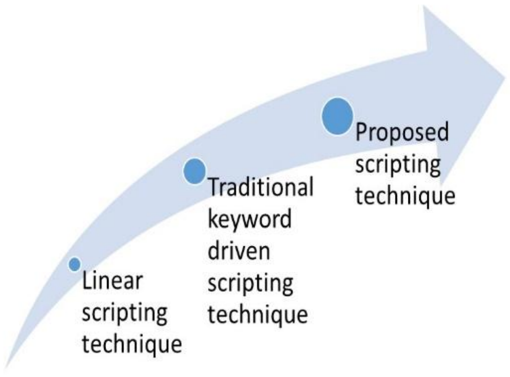
Linear scripting technique
Traditional keyword driven scripting technique
Proposed scripting technique

Figure 3. Moving Towards Higher Levels of Automation

In the next subsections, we elaborate on the features provided by the proposed framework.

A. Full or Partial Automation

Using any of the traditional scripting techniques (linear, data-driven, keyword-driven…etc.), the tester has to implement the complete test script manually from scratch [11]. However, using the proposed SAT framework, the tester does not need to perform this time-consuming task. To automate any test case using the proposed SAT framework, there are two major steps. The first step is to generate the TCS sheet. The second step is to run the proposed SAT framework on this generated TCS sheet to generate their corresponding test script.

The first step can be performed either manually or automatically. Therefore, our proposed SAT framework has two main running modes for its first step. In the partially automated mode, the tester has to list the test case steps manually in the high level proposed TCS sheet in a step by step manner. This will cost less time than filling down the complete sheet in its traditional format as this new format does not require any technical details. In the fully automated mode, the tester should only feed the SAT framework with the web page URL of the SUT and then the framework will auto-generate the test case steps in the TCS sheet without any user intervention. This will cost less time than the partially automated mode as test cases steps here is totally automatically generated and no more need for editing by the tester as in the partially automated mode.

The test script development task is the main difficulty in traditional automation approaches as it consumes a lot of effort. Therefore, the proposed SAT framework overcomes this problem by automating the second step. After the first step (performed either manually or automatically) is finished, the proposed SAT framework will automatically generate the complete test script for the test case steps in the sheet. Afterwards, this test script will be used to actually test the SUT.

Users in both partially and fully automated modes do not need to bother about searching for HTML input controls and their corresponding keyword and data.

The auto-generated test script does not have any logic statements such as If, For, While, Switch, Array…etc. Therefore, the testing engineer may modify this auto-generated test script. The fully automated mode can be used to stabilize a beta version of the SUT to be available for experienced users.

B. Proposed Framework Can Propose Data Values for Each Input HTML Tag

Since a higher automation level is sought, our proposed SAT framework does not only generate technical data in TCS sheet but it can also auto-propose appropriate data values based on HTML control input type. For example, if the object type is textbox, then the SAT framework proposes to fill in this input control with the value “test”. If the object is dropdownlist, then it proposes to fill it with the value “2”. To achieve this, there is a repository JSON file, which has a list of all possible HTML control types and for each type, there is a list of proposed major possible values that can fit to this input control type. The framework engine uses this JSON during auto-generating test case steps. This repository is editable by the tester. In addition, the proposed SAT framework is intelligent enough not only to propose values for HTML controls but also to detect validations and propose suitable positive and negative values, which match regular expressions for each HTML field. For example, if there is a textbox which is supposed to be filled in with an email address, the SAT framework will propose a list of values that will represent different positive and negative samples of the email address format. The same applies to other input data fields such as mobile numbers, phone numbers and credit card numbers.

C. Compatibility with All Browsers

Since not all web applications are compatible with all browser types, there is a need for a browser independent framework. Unlike Selenium IDE which is compatible with Firefox only, the proposed SAT framework is compatible with all browser types because:

- It depends only on the web page URL of the SUT to generate both the TCS sheet and its corresponding test script.
- It is implemented as a Windows application not as a specific browser plugin (such as Selenium IDE).

The proposed SAT framework is not only compatible with all available browsers, but also the auto-generated test scripts from our proposed SAT framework are able to run on any web browser without any test script modification. It generates the test script for the three main browsers; Internet Explorer, Firefox, Google chrome. Test scripts generated from Selenium IDE are tightly coupled to Firefox and hence the testing engineer has to update the test scripts to be able to run the script on both IE and Chrome web browsers. However, scripts generated from SAT framework are designed in such a way that depends on the abstraction concept. Therefore, if the generated test scripts need any modifications, due to requirements change for example, these modifications will be applied only once, and other browsers can also automatically read the updated scripts.

To achieve this important feature, our SAT framework is designed to implement Factory Design Pattern [34] as shown in Fig. 4. The BrowserFactory class will generate an object of type IBrowser based on a given string value. Thus, the framework can generate test scripts, which are totally independent of the web browser type.

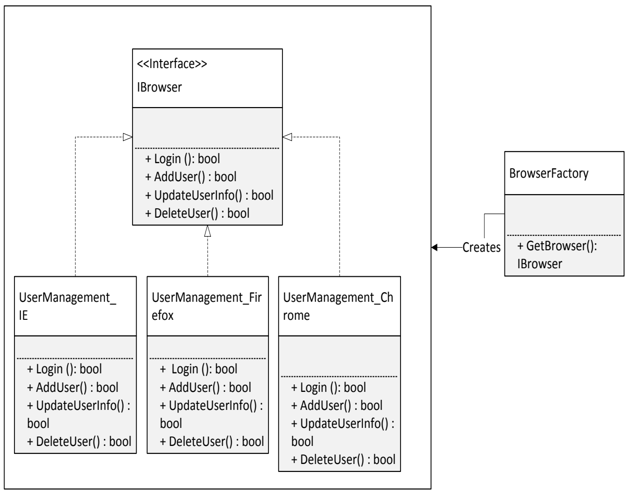

Figure 4. Proposed Class Diagram Snapshot for Multi-Browser Feature

D. *No Need to Run Tests Manually*

The traditional record/playback approach which is the base of many automation tools [26] [35] always requires the tester to run test case steps manually on the SUT then the automation tool will automatically generate the test script which simulates user actions. The tester is able to playback this generated script later.

Another advanced step in the record/playback approach was achieved in Selenium IDE. It asks the user to record scenario steps, then it generates well-structured programming test scripts. This leads to the generation of test scripts which are maintainable (unlike record/playback tools).

Our SAT framework has the ability to auto-generate maintainable structured programming test scripts without even recording test cases manually. It detects all HTML input controls and depends on them to auto-generate the script as shown in Fig. 5. It only needs the web page URL as an input, and then it automatically generates the corresponding test scripts and TCS sheet.

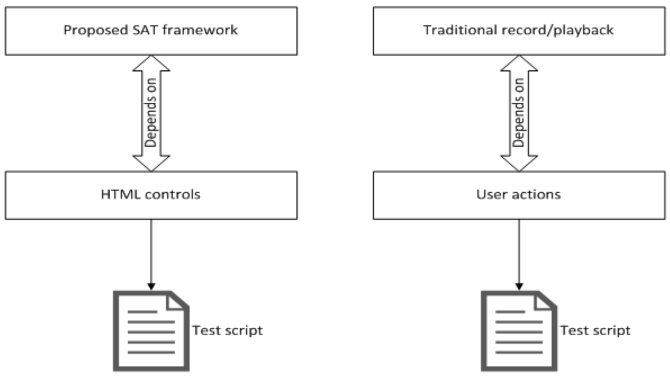

Figure 5: Proposed SAT framework Vs Traditional Dependency

E. *No Tight Coupling to a Specific Automation Tool*

In spite of that each automation tool has its own implementation details, the proposed SAT framework is designed to be generic enough to be plugged in any software automation tool. This means that the generated output test scripts from the proposed SAT framework is not tightly coupled to a specific automation tool such as Selenium, Watin,… etc.

This is achieved in SAT architecture by adding a new layer between the framework and the automation tools that the framework will integrate with. This middle layer is independent of the software automation tools (such as those mentioned in Table 1). Hence, if a new testing automation tool appears, all what is needed is to integrate our proposed SAT framework by implementing the proposed framework interface layer using this new automation tool syntax.

CASE STUDY

The study conducted by Garousi and Mesbah [36] reported that two major challenges face researchers during evaluation of automation approaches:

- Not all research papers conduct their empirical evaluation on the same web applications set. This results in a major difficulty during comparing testing techniques. To our knowledge, this problem has not being solved yet. Comparison would be easier if there is a predefined pool of web-based applications that is available for all researchers.

- Another difficulty is that very little research has been conducted for empirical evaluations on large-scale industrial web applications, most of the research in this area conduct their evaluation on small-scale web applications.

The above-mentioned shortages must be considered during selecting a web application to apply our SAT framework on. This section shows a complete case study for applying our proposed SAT framework on a real web application, namely “Subscription and Sales Management” modules. These modules have been selected since they are a basic entry point for most web applications. Any user opening most web applications is usually requested to login. If the user does not have an account, then he/she is requested to sign up (create a new profile). In later stages, the user may update or delete account data. Then the user will use his/her account to make action on the SUT. One of the most common scenarios in our case study is that the user is requested to create a new profile then login, search, select items to buy, confirm order and follow-up on order status. It is a highly beneficial to automate these necessary generic modules since testers are always asked to test the main critical functionality of the SUT after any new deployment even if the deployment does not affect these main modules. We created twenty test cases to test our proposed SAT framework. Table 2 shows sample of the test cases with their description.

Table 2: Sample of the Test Cases

Test Case ID	Description [This test cases is used to test that ....]
Test Case 1	User can login with valid credentials
Test Case 2	User cannot login due to invalid credentials
Test Case 3	User can create new account
Test Case 4	Admin can approve account details
Test Case 5	Admin can reject account details

After generating the test scripts, they should be run over the SUT [37]. During execution, the framework is responsible for reading test data from the external data source.

EXPERIMENTAL RESULTS AND DISCUSSION

To evaluate the effectiveness of the proposed framework, time needed using both traditional automation and Selenium IDE are compared to our proposed framework:

- Traditional Automated Testing: the tester implements test scripts from scratch using the traditional keyword-driven scripting technique.
- Selenium IDE Automated Testing: the tester implements test scripts using Selenium IDE.
- Proposed SAT Framework: the tester uses the proposed framework to generate both the test case steps and its corresponding test script by providing the framework with the web page URL to be tested.

Results of the time spent to test the same test case using the three testing methods are listed in Table 3. These numbers include only the time to develop testing scripts to be ready for the execution phase. All numbers are in minutes.

Table 3. Time Needed for Checklist Testing (in Minutes)

Test Case	Duration of Manual Testing	Duration of Automated Testing						
		Traditional Automation	Selenium IDE			Proposed SAT Framework		
		Scripts Development Time	Scripts Development Time	Inject Business Logic Time	Total	Scripts Development Time	Inject Business Logic Time	Total
Test case 1	20	20	3	0	3	1	0	1
Test case 2	20	20	3	0	3	1	0	1
Test case 3	40	30	6	5	11	2	5	7
Test case 4	15	15	3	1	4	1	1	2
Test case 5	15	15	13	1	14	1	1	2
Test case 6	30	19	5	4	9	2	4	6
Test case 7	19	20	2	5	7	1	5	6
Test case 8	14	30	3	4	7	1	4	5
Test case 9	10	16	4	2	6	1	2	3
Test case 10	15	34	5	0	5	1	0	1
Test case 11	60	100	7	20	27	5	20	25
Test case 12	10	40	3	10	13	2	10	12
Test case 13	20	40	5	10	15	2	10	12
Test case 14	10	30	3	12	15	1	12	13
Test case 15	17	40	1	16	17	2	16	18
Test case 16	8	30	5	6	11	1	6	7
Test case 17	23	28	3	3	6	2	3	5
Test case 18	10	30	1	8	9	2	8	10
Test case 19	90	140	11	30	41	10	30	40
Test case 20	19	18	2	5	7	1	5	6
Total	465	**715**	88	142	**230**	40	142	**182**

Table 3 shows that using the proposed SAT framework is better than using traditional automation and Selenium IDE approaches. The total time consumed in applying traditional automation techniques is approximately 715 minutes for intermediate skill testers (100% of the total testing time). The total time consumed in applying automation using Selenium IDE is approximately 230 minutes (≅32% of the total testing time). On the other hand, the total time consumed using the proposed SAT framework is 182 minutes (≅25% of the total testing time &≅79% of the Selenium IDE time).

Thus, using the proposed SAT framework can save approximately about 75% of the total time/effort involved in the automation process using traditional automation and 21% compared to using Selenium IDE as shown in Fig. 6.

The overall automation cost (which is 182 minutes in our case study) is paid off when these scripts are executed repeatedly [8]. Each time the test scripts are executed automatically, the overall automation cost is reduced gradually.

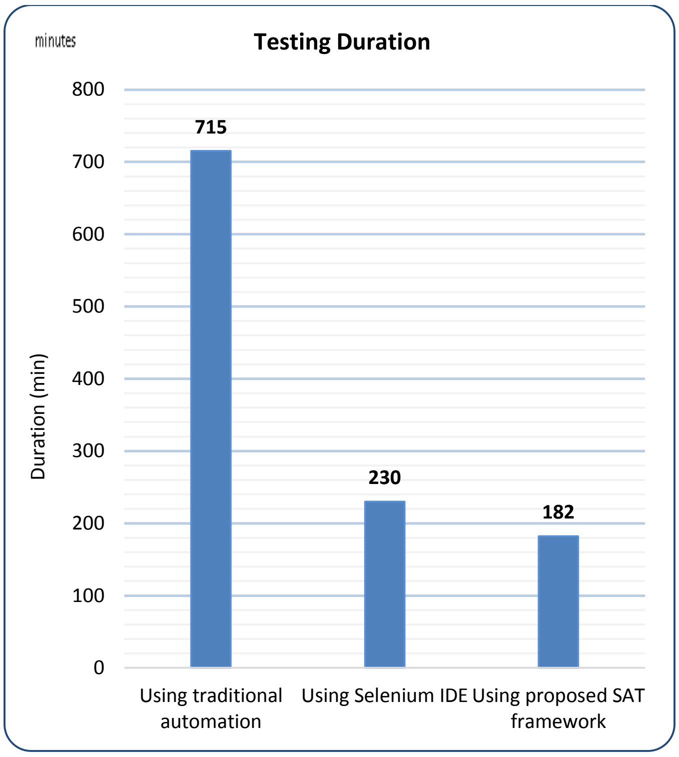

Figure 6. Duration Needed for CheckList testing

The total number of Lines of Code (LOC) which are implemented manually versus that generated automatically are compared in Table 4. Using the traditional automation approach, the testing engineer needs to implement all the test scripts manually from scratch which is 968 LOC (≅ 100% of the total LOC). However, using the proposed SAT framework, the tester has to implement only the SUT business logic of the test scripts which are approximately 92 LOC (≅ 10% of the total LOC) whereas the framework generates the other part of the test script (≅90% of the total LOC). Thus, using the proposed SAT framework can save approximately 90% of the total number of LOC.

Table 4. LOC Implemented by Tester and Framework

Automation Approach	Manually Implemented LOC	Automatically Generated LOC
Traditional Automation	968 (= 100%)	0
SAT Framework	92 (≅ 10%)	876 (≅ 90%)

The overall SAT framework performance is evaluated in terms of:

- Test Script Creation Time: The main target of our proposed SAT framework is to automate the test script development task. This is achieved by auto-generating a complete test project with **68%**-time reduction of the total time of the automation process.
- Usability: SAT framework provides testers with auto-generated high level TCS file instead of asking the tester to go deeply into technical details. This enables testers or business users to easily edit the TCS sheet as it does not require any programming background.
- Reusability: The generated output test project is built using the keyword automation testing technique. Therefore, this enables the testing engineers to easily create new keywords which are then used by the SAT framework.
- Maintainability: SAT framework generates the main and basic source code for testers. However, the testing engineer can easily navigate through the source code to inject some business logic for maintenance purposes (optional step) instead of creating source code from scratch. This step does not require the tester to have deep background in software development because the framework is built over an existing automation tool.
- Extensibility: Since SAT framework is not tightly coupled to a specific automation tool, this gives more flexibility to the tester to run the generated tests upon any new automation tool. However, this will require that the testing engineer implements the main keywords interface in the format of this new automation tool. Table 5 compares SAT framework performance to other tools [38].

Table 5. Comparison of Sat Framework with Other Sample Tools

	Record and Playback	**Cucumber**	**Robot Test Framework**	**Autotestbot**	**Proposed SAT Framework**
Maintainability	X	√	√	√	√
Free Form Test Cases	X	NA	NA	√	√
Reusability	X	√	√	√	√
Modularity Easy To Use	X	√	√	√	√
Consistency Test Case Execution	Low	High	High	High	High
Complexity For Non-Programmers	Easy to use	Complex	Medium	Easy to use	Easy to use
Browsers Compatibility	Browser dependent	NA	All browsers	Firefox only	All browsers

CONCLUSION AND FUTURE WORK

The primary objective of this research is to propose a new software automated testing framework which gives more help for testers in the overall software automation testing process. The proposed framework can successfully be used for automating the test scripts creation process. The proposed SAT framework is beneficial specially when the SUT is changing quickly and needs regression testing to confirm that the software version is stable. Also, testers or business users can do any other activity while running the test scripts on the SUT.

Experiments are conducted to assess the effectiveness of the proposed framework by applying the proposed framework on a case study. Results of these experiments are promising as it has proved that it can save nearly 68% of the overall software testing automation process and therefore shorten the product launch cycle. At the same time, it can achieve an amount of work that manual tests are almost difficult to finish. The proposed approach also generates the test cases into a standard tabular format which is easier and stricter than writing them manually in normal English language.

There are multiple directions for future work such as to find more generic methods to gain more enhancements in the overall testing effort to get better results and achieve a higher level of automation by auto-generating more abstract scripts.

REFERENCES

[1] S. Thummalapenta, S. Sinha, N. Singhania and S. Chandra, "Automating Test Automation," *34th International Conference on Software Engineering (ICSE)*, 2012.

[2] T. Kanstrén, "A Review of Domain-Specific Modelling, Software Testing," *The Eighth International Multi-Conference on Computing in the Global Information Technology*, 2013.

[3] A. Jain and S. Sharma, "An Efficient Keyword Driven Test Automation Framework for Web Applications," *International Journal of Engineering Science & Advanced Technology*, vol. 2, no. 3, pp. 600-604, 2012.

[4] S. H. Trivedi, "Software Testing Techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 10, pp. 433-439, 2012.

[5] V. Sangave and V. Nandedkar, "Generic Test Automation," *International Journal of Science and Research (IJSR)*, vol. 4, no. 7, 2015.

[6] P. Yalla, L. S. S. Reddy, M. Srinivas and T. M. Rao, "Framework for Testing Web Applications using Selenium Testing Tool with Respect to Integration Testing," *International Journal of Computer Science and Technology*, vol. 2, no. 3, pp. 165-170, 2011.

[7] M. Sadiq and F. Firoze, "A Method for the Selection of Software Testing Automation Framework using Analytic Hierarchy Process," *International Journal of Computer Applications (0975 – 8887)*, 2014.

[8] A. Cervantes, "Exploring the Use of a Test Automation Framework," *IEEE Aerospace conference*, 2009.

[9] John A. Clark, Haitao Danb, Robert M. Hierons, "Semantic mutation testing," *Science of Computer Programming 78 (2013)*, pp. 345–363,, 2013.

[10] O.-P. Puolitaival, T. Kanstrén, V.-M. Rytky and A. Saarela, "Utilizing Domain-Specific Modelling for Software Testing," *The Third International Conference on Advances in System Testing and Validation Lifecycle*, pp. 115-150, 2011.

[11] M. S. M. Yalla, "Building automation framework around open source technologies," *Proceeding of Software Testing Conference, Banglore*, 2009.

[12] H. S. Gill, M. Kaur, A. Puri and A. Jaswal, "Automation Framework of Browser Based Testing Tool Watir," *International Journal of Computers and Distributed Systems*, vol. 1, no. 3, 2012.

[13] V. Sangave and V. Nandedkar, "A Review on Automating Test Automation," *International Journal of Advance Research in Computer Science and Management Studies*, vol. 2, no. 12, 2014.

[14] S. Lamba, V. Rishiwal and A. Rana, "An Automated Data Driven Continuous Testing Framework," *International Journal Of Advanced Technology In Engineering,* vol. 3, no. 10, 2015.

[15] N. Javvaji, A. Sathiyaseelan and U. M. Selvan, "Data Driven Automation Testing of Web Application using Selenium," *CONFERENCE PROCEEDINGS, STEPAUTO* , pp. 32-37, 2011.

[16] F. Wang and W. Du, "A Test Automation Framework Based on WEB," *IEEE/ACIS 11th International Conference on Computer and Information Science,* pp. 683-687, 2012.

[17] A. Pillai, "Designing Keyword Driven Framework mapped at Operation Level," 2017. [Online]. Available: http://www.automationrepository.com/2012/08/keyword-driven-framework-mapped-at-operation-level-part-1/.

[18] M. Leotta, D. Clerissi, F. Ricca and P. Tonella, "Capture-Replay vs. Programmable Web Testing," *20th Working Conference on Reverse Engineering (WCRE),* 2013.

[19] V. G. Yusifoglu, Y. Amannejad and A. B. Can, "Can Software test-code engineering: A systematic mapping," *nformation, Software Technology* , vol. 58, p. 123–147, 2015.

[20] Jussi Kasurinen, Ossi Taipale, Kari Smolander, "Software Test Automation in Practice Empirical Observations," *Advances in Software Engineering - Special issue on software test automation,* 2010.

[21] T. Bharti and V. Dutt, "Functionality Appraisal of Automated Testing Tools," *International Journal of Computer Science Trends, Technology (IJCST),* vol. 3, no. 1, pp. 129-134, 2015.

[22] I. Singh and B. Tarika, "Comparative Analysis of Open Source Automated Software Testing Tools: Selenium, Sikuli, Watir," *International Journal of Information and Computation Technology,* vol. 4, pp. 1507-1518, 2015.

[23] K. Metha, S. Chavan and M. Patil, "Towards Developing a Selenium Based GUI Automation Test Pro: a Comparative Study," *International Journal of Engineering Research & Technology (IJERT),* vol. 3, no. 2, 2014.

[24] "Test Automation for Web Applications," 2017. [Online]. Available: http://www.seleniumhq.org/.

[25] "HP QuickTest Professional Tutorial," 2017. [Online]. Available: https://www.tutorialspoint.com/qtp/.

[26] H. Kaur and G. Gupta, "Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional, Testcomplete," *Journal of Engineering Research and Applications,* vol. 3, no. 5, pp. 1739-1743, 2013.

[27] Meenu and Y. Kumar, "Comparative Study of Automated Testing Tools: Selenium, SoapUI, HP Unified Functional Testing, Test Complete," *Journal of Emerging Technologies and Innovative Research (JETIR),* vol. 2, no. 9, 2015.

[28] M. Monier and M. M. El-mahdy, "Evaluation of Automated Web Testing Tools," *International Journal of Computer Applications Technology and Research,* vol. 4, no. 5, pp. 405-408, 2015.

[29] A. M. F. V. d. Castro, G. A. Macedo, E. F. Collins and A. C. Dias-Neto, "Extension of Selenium RC Tool to Perform Automated Testing with Databases in Web Applications," *8th International Workshop on Automation of Software Test (AST),* 2013.

[30] A. Ema and E. M. Reddyb, "Software Test Automation: An algorithm for solving system management automation problems," *International Conference on Information, Communication Technologies (ICICT),* vol. 46, pp. 949-956, 2015.

[31] T. R. Devi, "Propose Automated Software Testing Tools to Test Given Application Report Bugs," *International Journal of Engineering Research and Technology (IJERT),* vol. 2, no. 1, 2013.

[32] T. Kosa, M. Mernikb and T. Kosarb, "Test automation of a measurement system using a domain-specific modelling language," *Journal of Systems and Software,* vol. 111, p. 74–88, 2016.

[33] A. K. Jha, "Development of Test Automation Framework for Testing Avionics Systems," *29th Digital Avionics Systems Conference (DASC),* 2010.

[34] "Factory Design Pattern," 2017. [Online]. Available: https://www.tutorialspoint.com/design_pattern/factory_pattern.htm.

[35] A. Jain, M. Jain and S. Dhankar, "A Comparison of RANOREX, QTP Automated Testing Tools, their impact on Software Testing," *International Journal of Engineering, Management & Sciences (IJEMS) ISSN-2348 –3733,* vol. 1, no. 1, 2014.

[36] V. Garousi, A. Mesbah, A. Betin-Can and S. Mirshokraie, "A Systematic Mapping Study of Web Application Testing," *Information and Software Technology,* vol. 55, no. 8, pp. 1374-1396, 2013.

[37] M. Hammoudi, G. Rothermel and P. Tonella, "Why do Record/Replay Tests of Web Applications Break?," *IEEE International Conference on Software Testing, Verification and Validation (ICST),* 2016.

[38] A. Madhavan, "Semi Automated User Acceptance Testing using NLP," *Digital Repository Iowa State University,* 2014.