International Journal of Soft Computing 12 (3): 148-155, 2017
ISSN: 1816-9503

A Comparative Study of Testing Framework with Special Emphasis on Selenium for Financial Applications

Koneru Srinivas and Chandra Prakash
Department of Computer Science, KL University, Vijayawada, Andhra Pradesh, India

Abstract: Selenium and appium are the best choices of open source tools for testing web based and mobile based applications. For its cross platform and multi browser compatibility and its support to write scripts in C#, Ruby, Java, Python and Perl has gained its popularity. Since, financial and banking applications require the best testing solution for checking the functionalities, securities, compatibility, performance, stress and load with minimal infrastructural cost, selenium and appium with a hybrid framework would be optimum choice.

Key words: Web testing with selenium tool, selenium variants, selenium framework, selenium for financial applications, challenges and mitigations in testing of banking applications, mobile automation, estimating testing effort

INTRODUCTION

Selenium was first developed and was called as Java Script Test Runner at Chicago by Jason Huggins while working for thought works. This was created to test their internal time and expenses application written in Python programming language. Selenium is a chemical element in the periodic table that has the symbol Se with atomic number 34. Selenium comes from a joke made by Huggins in an email, mocking a competitor named Mercury that developed a tool called quick test professional, saying that mercury poisoning can be cured by taking selenium supplements.

Thought works is an IT outsourcing company that follows agile model for their software developments and this company has contributed many open source products among which selenium is one of the popular product. Huggins team at thought works got strengthened when more programmers and testers joined them and it was Paul Hammant who steered the development of the second mode of operation that later became Selenium Remote Control (RC). This tool was also released as an open source the same year.

Dan Fabulich and Nelson Sproul taking the help from Pat Lightbody transformed Selenium-RC by adding series of patches. Later Huggins joined Google and with Jennifer Bevan continued with the development of Selenium RC providing more stable releases.

Simon Stewart at though tworks developed a superior browser automation tool called webdriver and that became a boon for the automated QA testers as it used to interact with the browsers directly.

Philippe Hanrigou while working for thought works developed another tool called selenium grid which provides a hub to run concurrent selenium tests on any number of local or remote systems that minimizes execution time. Selenium grid is also an open source having capabilities similar to the internal or private google cloud for selenium RC.

Selenium is one of the most powerful open source automation tools for testing web based applications. Supports all the prominent browsers on almost all the operating systems since it was developed using Java Script, DHTML and frames. Automation scripts written for firefox for windows can be executed on firefox running on macintosh operating system.

The developers after a meeting at the google test automation conference merged selenium RC with webdriver and named it as Selenium 2.0 (2009). Selenium is one of the best tools available in the market for testing any kind of web based application considering its platform and cross browser compatibility. Selenium tool is definitely a boon for its users as it is very much customizable for the requirements.

Advantages of selenium tool:

- Open Source tool for Web applications
- Supports testing system functionality and browser compatibility
- No licensing fee
- As per requirements can be customized
- Compatible to different platform and browsers
- Works with different languages like C#, Ruby, Java, Python, Perl, etc.

Corresponding Author: Koneru Srinivas, Department of Computer Science, KL University, Vijayawada, Andhra Pradesh, India

- HTML is the default format
- Selenium scripts written for firefox on windows could be taken to run on firefox in mac

Selenium supported browsers

Windows operating system:

- Internet explorer
- Firefox
- Google chrome
- Seamonkey
- Opera
- Html unit
- Phantomjs

Mac OS X:

- Safari
- Firefox
- Camino
- Seamonkey

Linux operating system:

- Firefox
- Konqueror

Mobile supported browsers:

- Android (with selendroid or appium)
- iOS (with ios-driver or appium)

Selenium scripts runs with firefox as the default browser and to invoke other browsers a corresponding driver file will be required

Selenium variants: Selenium comes in four variants which could be used solely or in combination to create complete automation suite for testing web applications

- Selenium IDE
- Selenium core
- Selenium remote control and selenium webdriver
- Selenium grid

Selenium IDE: It was developed by Shinya Kasatani, Japan. It provides an integrated development environment to build selenium test cases. It is available as a firefox add-on and provides an interface for developing and executing individual test cases or an entire test suite. Using recording option all the user actions are recorded as they are performed and stored as a reusable script to play back. A set of selenium instructions used to test web application is called as Selenese. With its intellisense feature allows the user to pick from a list of assertions and verifications for the selected element or location. Even if it is a firefox add-on, tests created in it can also run on other browsers by using selenium-RC and specifying the name of the test suite on the command line. The recorded actions can be converted into the popular languages like Java, C#, Perl, Ruby, Python, etc. Advanced test scripts cannot be executed which can be noted as one of the limitations of Selenium IDE. This tool became a part of selenium package in the year 2006 (Anonymous, 2017).

Selenium remote control: This test tool was written in Java which allows user to write automated test scripts for web application developed in any programming language like Java,. NET, Perl, Python, Ruby, etc., selenium RC took the place to overcome the limitations of selenium IDE. Selenium RC helped the automated testers in making the test processing much easier. It provided an Interface and library to support languages like HTML, Java, C#, Perl, PHP, Python and Ruby. In this selenium server communicates directly with the browser using AJAX (asynchronous Java Script and XML). Get/post requests of HTTP (Hypertext Transfer Protocol) can send commands directly to the server. Selenium server acts as a client-configured HTTP proxy to stand in between the browser and website. Since the selenium server is written in Java, Java Runtime Environment (JRE) version 1.5.0 or higher is required to start the server.

Selenium webdriver: This tool was developed by Simon Stewart in the year 2006 at thought works. It was designed to provide a simpler browser automation tool to support dynamic web pages where elements of a page change without the page itself being reloaded. Webdriver was designed to overcome the limitations of selenium-RC that could not work for file upload or download process, pop-ups, dialogs, etc. Along with language bindings, implementation of the individual browser controlling code, well-designed object-oriented API was made into Webdriver tool to support advanced web applications. It utilizes browsers native compatibility to automation without using any peripheral entity. Selenium 1.0 with webdriver combination resulted Selenium 2 (Rattan, 2013).

Selenium grid: This tool was developed by Philippe Hanrigou where the test suite is setup such a way that multiple instances of the process are executed on various platforms in parallel. It reduces the time required for running acceptance tests to a fraction of the total time. A grid consists of a single hub configured with one or more nodes registered to it. Once the registered nodes have been selected the instructions from the test script are sent to the hub which is then passed to the node through the requested browser-platform combination for execution.

This way of running tests in parallel takes very less amount of time which is almost equal to the maximum time by any single test.

Selenium automation framework: Automation testing has reduced the effort in terms of time and cost drastically when compared to manual testing. With the advent of various automation frameworks, furthermore testing time reduced to a great extent. Each framework comes with a set of guidelines which increased automation efficiency thereby reducing the cost. It is associated with many benefits such as code reusability, reduced maintenance cost, higher portability, reduced test data set-up time, increases software quality and reliability. Selenium Webdriver offers various test automation frameworks to choose from. Depending on the requirement of any application, a proper framework must be installed.

Use of frameworks

Customizable: Frameworks can be customized by modifying across the projects.

Maintenance: The scripts maintenance is very easy. We must have a version control that maps the released version of source code with the corresponding set of acceptance tests.

Re-usability: Test scripts or test suites can be reused by other teams when integrating modules where data, object repository or keywords are in common.

Redundancy: Test framework helps in avoiding duplication of test cases.

Reduces complexity: Helps in building library of functions that hides underlying complexity from users.

Configurability: Test automation framework helps in setting up environmental configurations as per the requirements. It helps in rearranging the priorities based on the business impact and even for functional regression test suites.

Scalability: Ensuring scaling the acceptance tests thereby accelerating the feedback loop to verify or validate the software to decide its release.

Auditability: Automation frameworks records various logs to keep track of the test execution along with the time thereby giving us scope for a quick audit and monitor outcomes.

MATERIALS AND METHODS

Types of automation frameworks

Linear framework: This framework is mainly based on record and playback and that follows linear form of scripting which can be used for very small projects. It is created for performing smoke test suites where only basic tests are executed. Selenium IDE can be used to generate scripts by recording the application which is ideal example of this kind of framework.

Advantages of linear framework:

- Test scripts can be easily written and it consumes minimal time to write the scripts
- Since the entire flow of the scripts is written in a linear order it becomes very easy to understand and make changes
- Extensive planning is not required as the built-in functionalities are used

Disadvantages of linear framework:

- Lacks reusability
- Tough to maintain this kind of framework as the change in development effects all the relevant scripts
- Since the input data is hard-coded, test cases cannot be executed with multiple data sets
- If the application is modified at any given point, a lot of rework is required as the same type of code is replicated at several instances

Data-driven framework: This framework helps eliminating the hardcoded data and the data is inputted and the expected output results are stored in a separate data file (normally in a tabular format) so that a single driver script can execute all the test cases with multiple sets of data. The driver script contains navigation through the program, reading of the data files and logging of the test status information.

Advantages of data-driven framework:

- A single test can be executed with multiple data sets
- Hard-coding can be avoided
- Faster execution helps in saving time
- Well defined architectural design
- Script execution in multiple environments
- Very efficient in the analysis of result logs
- Communication of results
- Easy debugging and script maintenance
- Robust and stable due to error and exception handling

Disadvantages of data-driven framework:

- Strong technical expertise is required to identify external data sources and create functions that connect to these sources seamlessly

Key word driven framework: Key word driven test is a kind of functional automation testing where use cases are written as key words in an external file like excel spreadsheet. These key words serve as guides as to what action needs to be performed on the application. Basic understanding of the workflow is enough to maintain the tests since all the required functions and operations are written in an external file. This framework reduces the scripting effort of the tester. At the time of execution the driver script interprets the keywords and its associated action from the excel spreadsheet and then executes the instruction. This framework fits to any automation tool even when there is a change in the test automation tool as we need a new driver script infrastructure to interpret and execute keywords and test scripts. It is an extension of the data driven testing as it segregates test data from test scripts (Singla and Kaur, 2014).

RESULTS AND DISCUSSION

Page object: In this framework an object repository is created for the web elements or objects on the application. This really helps when the objects definitions or html identifiers in an application keeps changing. For each page of a web application the objects or classes are identified and defined in the object repository.

Advantages of page object pattern:

- If HTML identifier changes for an element, changing it in the page-objects will be enough as the test scripts reads from this centrally located repository
- Key words also can be defined along with the object definition
- Page navigation could also be defined

Testing framework (test next generation): It is another power automation testing framework developed by cedric beust that evolved from JUnit (Java Unit) and NUnit (Net Unit). It provides some predefined annotations that could be used in prioritizing the tests. When we build large test suites with many test cases, sometimes it would become cumbersome for us to identify the order in which it needs to execute. And more over when the sequences of links or modules are altered in the application then it becomes even more difficult to rearrange. For automated testers this comes like a wonderful framework that solves the issue by just placing the required annotation that should go in order (Gaur and Chhillar, 2012).

Modular testing framework: To build this framework first we must identify the repetitive functionalities and then build a script such a way that it is called from different test suites thereby reducing the testing effort, time and cost. The major functionalities are divided into smaller modules so that test scripts could be built easily and these test scripts are combined together to build a larger test script when handling a larger module.

Behavior driven testing framework: This framework allows the automation of functional validations in a simple and easy understandable format for the developers and testers. In this kind of frameworks programming knowledge is not required.

Hybrid framework: The hybrid automation framework is created by combining distinct features of two or more frameworks. Hybrid-driven framework is a collection of two or more frameworks that can be customized and accessed by any user. For example, a combination of page objects, a key word-driven framework, a data-driven framework an object repository and reporting listeners provides a powerful hybrid framework. This helps to leverage the benefits of all the frameworks.

Advantages of hybrid framework:

- The Hybrid Framework is a blend of the best features of various automation frameworks
- It is highly robust and flexible if properly implemented

Disadvantages of hybrid framework:

- Strong technical expertise is required to design and maintain the hybrid framework

Test automation plan for financial applications: Financial services and banking sectors have been bridging with the new technologies to leverage the optimization of the functionalities. These industries have taken a leap with digital transformation for achieving their business goals while enhancing customer experience. Banking and financial applications are designed with web and mobile interfaces that have largely improved customers experience. People these days are using plastic money which refers to the credit card, debit card, pre-paid cash cards, store cards, etc. has become more convenient method in making transactions for shopping, travel, etc. So, banking and financial sectors are encouraging mobile wallets, P2P transfers, omni channel banking, etc. The biggest concern of banking and financial sectors has been the security since the financial functionalities has been intertwined into every commercial application. With the

evolution of mobile commerce in 1997, online marketing and sales took off wherein financial services industries are playing a key role. Digital transformation brought a great change to the society but at the same time is very much vulnerable to security threats. Banking services are getting updated with innovative design and technology to satisfy their tech savvy customers. Since these new processes exchange sensitive business information it becomes a critical task in ensuring secure functioning of the system. Testing of such applications would become a great challenge for the QA teams. Cross platform, device compatibility have always remained the main concern of any banking or financial sector.

Repeating functionalities could be segregated thereby applying modular framework and since these applications require huge amounts of data inputs a data driven framework could be used. The ideal framework for leveraging the benefits of automation testing on any banking or financial application would be a mix of various frameworks the hybrid testing framework.

After a thorough cost-benefit analysis, open source automation tool like Selenium was found an adaptable and robust solution for banking and financial applications.

Banking domain: Banking domain is characterized into two sectors.

Traditional banking sector:

- Core banking
- Corporate banking
- Retail banking

Service based banking sector:

- Core banking
- Corporate banking
- Retail banking
- Loans
- Trade finance
- Forex
- Private banking
- Consumer finance
- Customer delivery channels

Characteristics of a banking application: Banking application emphasizes on key factors, based on which the test strategy needs to be built. A standard banking application must have these characteristics, presented below:

- Must support concurrent users
- Integrating with other applications like trading accounts, utility payments, credit cards, cash loading, etc.
- Transactions must be carried out with good pace
- Huge data storage is a must
- Data security must be tight
- Capable for audits
- Handling complex business workflows
- Supporting different operating systems like Mac, Linux, Unix, Windows, etc.
- Supporting users from multiple locations
- Supporting multi-lingual users
- Supporting users with various payment systems like VISA, MasterCard, etc.
- Supporting multiple service sectors like loans, retail and corporate banking
- Deployed with disaster management system

Challenges in testing the applications of banking domain: Digital transformation is on the radar of all the financial houses providing virtual services like anytime or anywhere banking. Being a customer centric shared platform, mobile application poses to be a big challenge. Cross browser testing has always been a great challenge as the users interact with different browsers like Safari, Opera, Firefox, IE, Chrome, etc.

Hackers mostly aim at banking and financial portals using SQL injections where security testing poses to be a big challenge. Maintaining international security standards such as BASEL II, BASEL III and Basel Committee on Banking Supervision (BCBS) 239 is something experts harp on. Money laundering, tax evasion, etc. are to be considered. So testing of financial applications becomes very important and challenging.

Performance failures will have a serious impact. Performance depends on the infrastructure, optimization, network connectivity and database integration. Stress and Load tests needs to be performed at regular intervals. Again another challenging task when creating test data from production data for performing data driven test. The biggest challenge in testing banking system is during the migration of the system from the old to the new one where the data will be uploaded and transferred to the new system after migration.

Integration testing poses to be another great challenge as each user of the system might access the application in a different sequence so the testing should ensure that each link connects to the other pages seamlessly.

Since, banking applications are used by wide strata of people and some of them lack necessary technical skills, so usability testing fall into place to ensure simple and efficient design to make it easy to use by various groups of customers.

Mitigation: Compatibility testing across different platforms must be performed to ensure smooth and easy operation of the application. Proper securities at each level must be implemented to ensure the system meets international compliances, standards and guidelines. Rigorous security checks need to be performed to ensure masking of input data, data encryptions, virtual keypads, integration with mobile phones in providing dynamic access code, captcha images, clearing sessions and cookies.

Regression testing should be implemented to ensure data migration is perfect. Parallel testing of old and new systems must be conducted to ensure the new system has all the data and functions better than the old one. Data driven framework must be implemented to ensure the system runs with huge amounts of data.

Performance testing plays a vital role in analyzing the bottlenecks of the system. So, proper checks must be implemented to validate data processing, database integration, page loading, browser connectivity, network connectivity, hardware support and concurrent users load. System integration tests must be conducted to ensure all the modules, libraries, databases, networks, hardware devices and external interfaces are intact.

Mobile automation framework: Using of smart phone applications has experienced tremendous growth and banking applications has made its space to a large extent. Mobile based banking applications are always a big challenge. So, there are various factors that needs to be considered while testing against mobile compatibility:

- Operating on various browsers
- On various operating systems
- Vendors skins
- Vendor technologies
- Geographical locations
- Network providers
- Data transfer

Various tests for user interfaces, functionality, security, usability, compatibility, performance, stress and load testing must be performed for the mobile applications. There are many automated test tools that could be used for testing mobile based applications and to name a few are listed below:

- Appium
- Calabash
- Robotium
- Selendroid
- Monkey talk or cloud monkey
- Espresso
- Touch test
- Egg plant or test plant

Appium is known to be the best open source tool that emerged to provide a cross-platform test automation framework for testing native, hybrid and mobile based web applications. It supports different languages like Java, Perl, C#, Python, Ruby and PHP in developing the test scripts. Accessing the source code or libraries is not required to conduct a test. It helps testing on Android and iOS operating systems. It facilitates configuring emulators that mimics the vendor mobile skin and technology. Appium supports remote execution and concurrent users testing on multiple devices.

Cost estimation: Proper analysis and effort estimation is necessary for successful planning of any testing project. Any flaw in critical estimation phase, results in missing the project deadlines, thereby increasing the cost of the project. Estimation can thus be considered as a technique which is applied when we take a proactive approach to the problem. Thus, estimation can be used to predict how much effort with respect to time and cost would be required to complete a defined task.

Estimation techniques used

Test point estimation: It is the simplest and easily understandable estimation technique that is widely used across the software testing spectrum. Iterative phases and simplicity are the most important features of this particular technique.

Work-phase based estimation: It is the estimation technique which is used whereby a guess estimate is made on a particular phase (normally the shortest and simplest of the phases) and then the testing team gradually adds on other phases into the initial estimation and finally comes up with an appropriate estimation.

Use-Case point estimation: It is the estimation on the use cases where the unadjusted actor weights and unadjusted use case weights are used to determine the software testing estimation.

Frameworks and its cost comparisons: Using of automation testing with other frameworks increased the maintenance costs for the customer. It also did not meet the customer's demand for a reusable and flexible testing framework and resulted in increasing costs and reducing efficiency of the testing framework.

After a thorough cost benefit analysis, I suggested a hybrid model to design, develop and deliver end-to-end automated testing solution for the customer. I am confident of my solution which would reduce the maintenance costs while improving the flexibility of the entire framework.

Estimating testing effort: Time estimation is calculated for the hybrid framework model using use-case point estimation for a banking application. Testing effort cost ($TC_{test\ effort}$) is calculated by the number of cycles of iterations of the testing period and acceptance testing period:

$$TC_{test\ effort} = (Numcycles_{test} \times TC_{test}) + (Numcycles\ A_{CCP} \times TC_{ACCP})$$

The banking application evaluated by me has 150 use cases with 70% complex use cases. For each complex use case average of 30 test cases are required and each complex test case takes 10 min to test. For each simple use cases average of 10 test cases are required and each simple test case takes 5 min to test. TC_{test} is determined by calculating the hours spent for test execution. Hours spent to test the entire system (TC_{test}):

$$= (150 \times 0.7 \times 30 \times 10) + (150 \times 0.3 \times 10 \times 5)$$
$$= 31500 + 2250$$
$$= 33750\ min$$
$$= 33750/60 = 562.5\ h$$

Average cycles for testing period is 20 and average cycles for acceptance testing could be 5. Therefore total cost for testing effort $TC_{test\ effort}$ is calculated as:

$$TC_{test\ effort} = (20+5) \times 562.5 = 14062.5\ h$$
$$TC_{test\ effort\ with\ automation} = 50\% \times 14062.5$$
$$= 7031.25\ h$$

Above calculated cost is the result of Linear Framework. We can further reduce the time by applying various frameworks. A detailed comparison is given below. The modularity testing framework is built on the concept of abstraction. This involves the creation of independent scripts that represent the modules of the application under test. Create customer account, update customer account, delete customer account, manage loans accounts, provident fund linking, transactions like deposits, withdrawals, inter-banking, fund transfers, fixed deposits, utility bill payments, report generations, user password maintenance are modularized so that number of testcases as compared to linear framework gets further reduced by 25%:

$$TC_{test\ effort\ with\ modular\ framework\ automation}$$
$$= (25\% \times 7031.25)$$
$$= 1757.8\ h$$

Now applying the data driven framework is applied where all the test input and the expected output results are stored in a separate excel data file so that a single driver script could execute all the test cases with multiple data sets. The driver script contains navigation through the functionalities into each module, reading the data files and logging the test status information. This further more reduces the test effort by 20% described as follows:

$$TC_{test\ effort\ with\ data\text{-}driven\ framework\ automation}$$
$$= (20\% \times 7031.25)$$
$$= 1406.25\ h$$

Key word driven testing is an application independent framework utilizing data tables and self-explanatory key words to explain the actions to be performed on the application under test. In this actions or key words are put in external input data file. Key word based testing is an extension to the data driven testing which further more reduces the testing effort with 5%:

$$TC_{test\ effort\ with\ keyword\ driven\ framework\ automation}$$
$$= (5\% \times 7031.25)$$
$$= 351.56\ h$$

Applying a combination of modular, key word and data driven on the banking application has reduced the test effort drastically thereby lessening the incurred cost. Account maintenance, customer maintenance, account types, report generations when modularized and test case actions were built as keywords and all the input data fields were parameterized and data was pulled from an excel file reduced the number of test cases:

$$TC_{test\ effort\ with\ hybrid\ framework\ automation}$$
$$= TC_{test\ effort\ with\ automation} -$$
$$(TC_{test\ effort\ with\ modular\ framework\ automation} +$$
$$TC_{test\ effort\ with\ data\text{-}driven\ framework\ automation} +$$
$$TC_{test\ effort\ with\ keyword\ driven\ framework\ automation})$$
$$= 7031.25 - (1757.8 + 1406.25 + 351.56)$$
$$= 7031.25 - 3515.61$$
$$= 3515.64\ h$$

Depending on deadlines of the project we can plan the number of resources required (Nageswaran, 2001).

CONCLUSION

Selenium is one of the best open source testing tools which supports different frameworks for testing web based applications. It has gained wide acceptance as a popular and successful mode of web and mobile applications testing. It's cross platform and cross browser compatibility is one of its best quality to consider. It's a boon for testing professionals to choose the language they are comfortable with to develop the test scripts. Selenium proved to be the most reliable and efficient tool when handling banking or financial applications as there are numerous things to consider like fast and secure transactions, massive database storage, support multi-lingual users, support users from multiple locations, support users on various payment systems like VISA, AMEX, master card, etc. and support users on multiple platforms like Mac, Linux, Unix, Windows, etc.

In study, this research demonstrates the comparison of different frameworks and the test automation tool which could be the best in terms of cost for any banking application. After a thorough cost benefit analysis, my research would suggest using Selenium Automation Toolwith a hybrid framework for the best results in terms of functionality, reliability, reusability and performance and the most important factor being the cost.

ACKNOWLEDGEMENT

I thank Dr. V. Chandra Prakash, Professor, Computer Science Department, K.L.University, Vijayawada Andhra Pradesh for his valuable guidance and support in completing this study.

REFERENCES

Anonymous, 2017. Testing banking & financial applications: Challenges, trends and best practices. Cigniti, India. http://www.gallop.net/blog/testing-banking-financial-applications/.

Gaur, D. and D.R.S. Chhillar, 2012. Implementation of selenium with JUNIT and test-ng. IJCSMS. Intl. J. Comput. Sci. Manage. Stud., 12: 226-232.

Nageswaran, S., 2001. Test effort estimation using use case points. Quality Week 2001, San Francisco, California, USA.

Rattan, R., 2013. Comparative study of automation testing tools: Quick test pro and selenium. VSRD. Intl. J. Comput. Sci. Inf. Technol., 3: 1741-1743.

Singla, S. and H. Kaur, 2014. Selenium keyword driven automation testing framework. Intl. J. Adv. Res. Comput. Sci. Software Eng., 4: 125-129.