A Novel method to generate constant and reusable test scripts and an Automated report generator for web UI

[1]Mahadevaswamy S, [2] Dr. Nirmala M. B

[1]MTECH Computer Network Engineering, [2]Associate Professor
[1]Department of Computer Science and Engineering
[1]Siddaganga Institute of technology, Tumkur, India.

Abstract : Graphical User Interface is one of the most used and user-friendly UI. The complexity of the graphical user interface is increased to the levels that; the user needs to be guided to use these UIs. Hence, testing of the UIs before the release, is one of the most important aspect in product release life- cycle. Testing of GUI is also a complex task. Manual testing will consume more time and error prone due to human errors. Hence, Automated testing is done to increase the speed of testing process and make the process error free. The GUI will be updated frequently until the arrival of the stable build. Due to the frequent updating of the UI, the test scripts need to be updated frequently whenever there is a change in the UI. This needs a solution. In our paper, we have proposed a novel method to reuse the test scripts even when the web user interface is changed. We are using a web user interface to test, that is used to configure the gateway product. We also propose an automated report generator which, gives a report of all the available dynamic-user interactable fields in a new web page; which helps the user in type of fields available and its uses like, type of values it accepts in a new web page. We are using Robot frame work tool which is python based. It is proved that, we can reuse the test scripts with minimal changes, even when the web UI is updated. With the use of both the proposed method and automated tool, increased the speed of testing and making the process free from human errors.

IndexTerms - GUI, web UI, Robot Framework, Manual Testing, Automated Testing, web element locators, static fields, dynamic user interactable fields.

I. INTRODUCTION

A software application which is more dynamic needs to have a user interface using which a user can handle the application and can make most out of the application using it. There are different types of user interfaces like Graphical user interface, Command line interface and others. The graphical user interface is the most popular user interface among the existing user interfaces. The operating systems like windows will use graphical user interface which is more user friendly and most popular also. Whereas operating systems like Linux will have command line interface where user as to know the command to use the application. But graphical user interface will remove this barrier.

The Graphic user interface has made the use of application more ease. But the development of UI is more complex. As the application gets complex, the user interface will equally get complex. One of the graphical user interfaces is web user interface; where the user interacts with the web server through the web user interface. The user interface will have Static fields and user interactable fields; where the user can interact with the underlaying application using these fields. The Graphical User Interface will play an important role in any application development, since it's the most promising way only through which the user can interact with the application. Without the proper working of the Graphical user interface the application will become incomplete even if the application is in its full potential.

So, it is necessary to test the User interface before the release of any application. Faults in the User Interface will break the connection between the user and the application. At days the Graphic User interface is becoming more complex and it will have several users interactable fields where each field needs to be tested for its proper functionality.

Manual approach of testing the web user interface is time consuming and cost ineffective. Manual testing involves human resource. In manual testing the tester as to test each field in graphical user interface. To do that, the tester must use each interactable fields in GUI and should check for its proper functioning. The tester must have the knowledge of each field like, field acceptance values and corresponding input and output parameters. The manual process is ever time consuming. The important aspect is that the manual process is more error prone. Hence, we need have automated framework for testing the user interfaces.

Automated testing involves many aspects. To build an automated test platform, the language on which the test platform is built plays an important role. The test architecture must be strong to drive the functionality test. The automated frame work must deal with the constantly updating user interfaces. The graphical user interface will be constantly updated for an application under test and needs to be tested frequently. The position of the fields in the user interface will be changed frequently until a constant and stable User interface is developed.

Hence, a proper approach is needed to handle the user interface. In the paper a novel method is described, which handles the testing of frequently changing fields in a web UI page. The paper also describes an automated tool that generates a report of interactable fields needs to be tested in a web user interface.

The web User Interface testing can be automated using many languages like selenium, python and open source framework like robot framework can also be used for automation. We are using robot frame work which is built on python platform.

II. Case Study.

In [1] the author, has described about the reusability of test scenarios based on the similarities of the web pages. The proposed method explains about finding the similarities in the GUI pages. Based on the similarities the authors are reusing the test cases. This method is good, for GUIs which have similar fields in them.

In [2] the author, describes about the use of X – paths; to locate the web elements. Authors have given an approach where they compare the Interface of a new version and the older version UI based on the difference between the DOMs of the two versions. DOM uses tree structure. Authors are finding the difference between the two pages of the UI using the DOMs tree structure. Using the addition or deletion of the node in the tree.

In [3] the author, have used the robot framework test automation generation. Robot Framework is built on python. The framework is key word driven. The frame work has huge set of libraries like selenium library for Web UI automation, AutoIT Library for Windows based application automation, SSH Library, Telnet Library, Built-in libraries and others. The test scripts can be easily written using the keywords. It produces the report Sheet in HTML format which is eases the bug detection.

In [4] the authors, have described a way to make the framework resistant to web UI changes. The GUI will have two types of fields. A dynamic field, using which the user can input his values. Examples are input type, radio type, buttons, checkboxes and others. A static field, where the value of these field will be either static or changed by the code itself and not by the user. Most of the static fields can be located using the X-paths. These X-path values changes for every change made to the GUI. But the approach only works for dynamic input fields and static fields needs x-paths which cannot be predicted. Hence the proposed method by the authors only works for dynamic fields.

As seen all the above proposed methods will only works on dynamic input fields which are interactive with the users. The static fields values will mostly depend on the x-paths. These x-paths are based on the position of the web elements. Even a small change in the UI like display of ‘error message’ will change the value of the x-path every time. So, it is necessary to take care of the static fields. The use of proposed method in our paper will have x-paths in to consideration. The approach considers both static and dynamic fields. Use of this approach will includes both x-path and other types of web element locators like name, id and others.

III. Proposed Approach.

3.1 A novel method to make web UI automated test scripts constant and reusable.

3.1.1 Work Flow.

In the proposed work we are using the robot framework. It is keyword driven and easier to write the test scripts. In this method we are separating variable parameters from the hard code. These variable parameters are stored in an excel sheet.

The algorithm flow: At first, we have to create the excel sheet using the format of ‘key:value’ pair. The first column will be filled with the Key names. Each name should be unique and non-repetitive. The second column takes the web element locators values and the continued columns will also take the web element locator values of different web pages.

The test scripts are coded to read the values from the excel sheet at execution and store it in a Dictionary. The Dictionary will store the values in the form of ‘Key:Value’ pairs. At every execution start, the script will read the key values from the first column of the excel sheet and assigns the corresponding value to each key word. At coding part, it is called using the key.

The example code line is like the following:

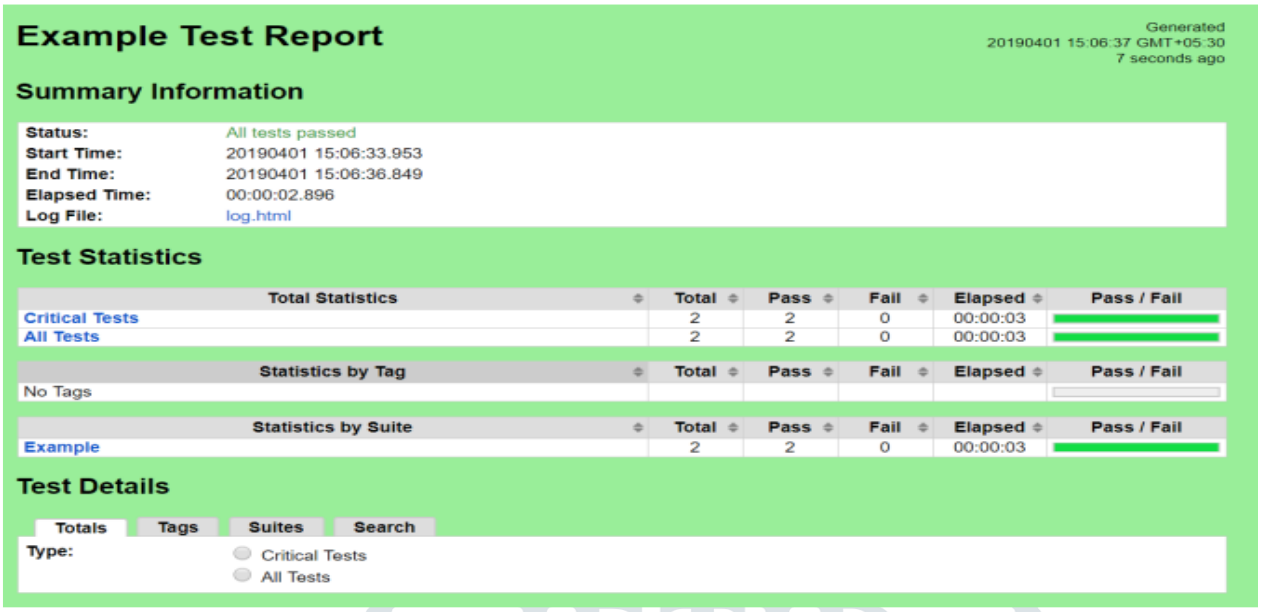
```
*** Test Cases ***
Startup script
    # reading identifiers from excel sheet and creating dictionary
    #and reading parameters from excel and making dictionary
    InitializeLocators
    ${VM1} =  SetConnection  ${HOSTVM}  ${USERNAMEVM}  ${PASSWORDVM}
    Set Global Variable  ${VM1}

Example
    Switch connection  ${VM1}
    Log  &{D1}[accessLocator]

    ${IP} =  IfConfig
    LOG  ${IP}
```

Sample code: Reading location values from Excel file.

The report of the robot framework which is showing read values of the web element locators from the excel sheet.



Example Test Report

Generated
20190401 15:06:37 GMT+05:30
7 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20190401 15:06:33.953
End Time:	20190401 15:06:36.849
Elapsed Time:	00:00:02.896
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	2	2	0	00:00:03	
All Tests	2	2	0	00:00:03	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

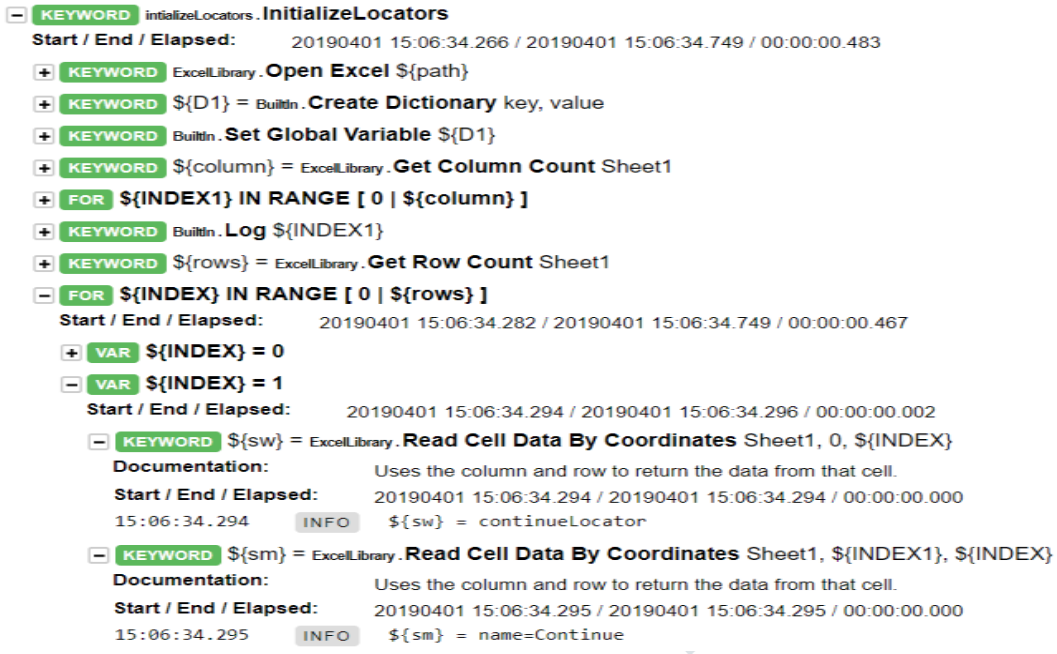
Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Example	2	2	0	00:00:03	

Test Details

Totals | Tags | Suites | Search

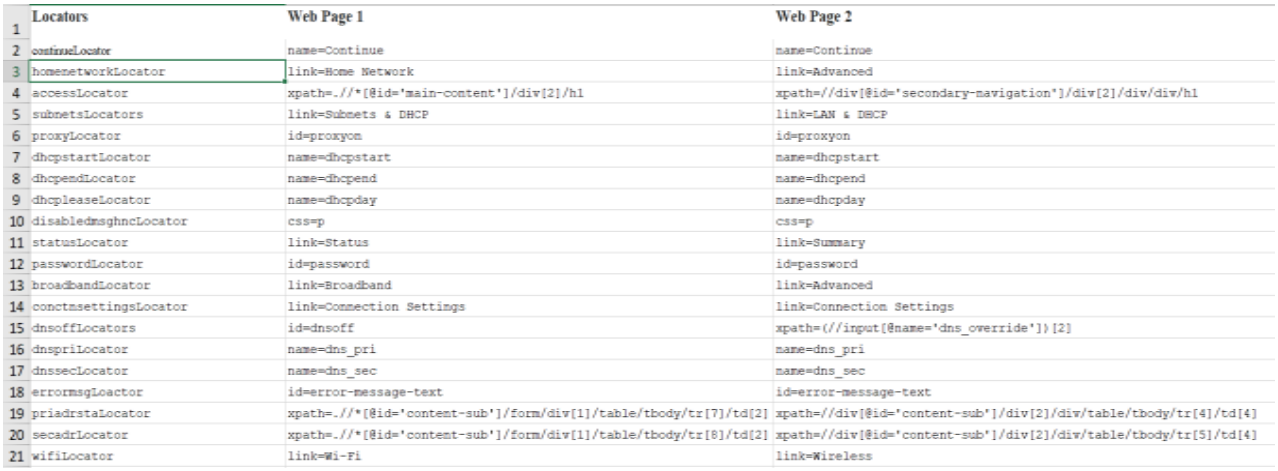
Type: Critical Tests / All Tests

Fig 1: Example test report (.html format)



KEYWORD intializeLocators . **InitializeLocators**
Start / End / Elapsed: 20190401 15:06:34.266 / 20190401 15:06:34.749 / 00:00:00.483
KEYWORD ExcelLibrary . **Open Excel** ${path}
KEYWORD ${D1} = BuiltIn . **Create Dictionary** key, value
KEYWORD BuiltIn . **Set Global Variable** ${D1}
KEYWORD ${column} = ExcelLibrary . **Get Column Count** Sheet1
FOR **${INDEX1} IN RANGE [0 | ${column}]**
KEYWORD BuiltIn . **Log** ${INDEX1}
KEYWORD ${rows} = ExcelLibrary . **Get Row Count** Sheet1
FOR **${INDEX} IN RANGE [0 | ${rows}]**
Start / End / Elapsed: 20190401 15:06:34.282 / 20190401 15:06:34.749 / 00:00:00.467
VAR **${INDEX} = 0**
VAR **${INDEX} = 1**
Start / End / Elapsed: 20190401 15:06:34.294 / 20190401 15:06:34.296 / 00:00:00.002
KEYWORD ${sw} = ExcelLibrary . **Read Cell Data By Coordinates** Sheet1, 0, ${INDEX}
Documentation: Uses the column and row to return the data from that cell.
Start / End / Elapsed: 20190401 15:06:34.294 / 20190401 15:06:34.294 / 00:00:00.000
15:06:34.294 INFO ${sw} = continueLocator
KEYWORD ${sm} = ExcelLibrary . **Read Cell Data By Coordinates** Sheet1, ${INDEX1}, ${INDEX}
Documentation: Uses the column and row to return the data from that cell.
Start / End / Elapsed: 20190401 15:06:34.295 / 20190401 15:06:34.295 / 00:00:00.000
15:06:34.295 INFO ${sm} = name=Continue

Fig 2: 'IntializeLocators' function which reads value from excel sheet.



	Locators	Web Page 1	Web Page 2
2	continueLocator	name=Continue	name=Continue
3	homenetworkLocator	link=Home Network	link=Advanced
4	accessLocator	xpath=.//*[@id='main-content']/div[2]/h1	xpath=//div[@id='secondary-navigation']/div[2]/div/div/h1
5	subnetsLocators	link=Subnets & DHCP	link=LAN & DHCP
6	proxyLocator	id=proxyon	id=proxyon
7	dhcpstartLocator	name=dhcpstart	name=dhcpstart
8	dhcpendLocator	name=dhcpend	name=dhcpend
9	dhcpleaseLocator	name=dhcpday	name=dhcpday
10	disabledmsghncLocator	css=p	css=p
11	statusLocator	link=Status	link=Summary
12	passwordLocator	id=password	id=password
13	broadbandLocator	link=Broadband	link=Advanced
14	conctnsettingsLocator	link=Connection Settings	link=Connection Settings
15	dnsoffLocators	id=dnsoff	xpath=(//input[@name='dns_override'])[2]
16	dnspriLocator	name=dns_pri	name=dns_pri
17	dnssecLocator	name=dns_sec	name=dns_sec
18	errormsgLoactor	id=error-message-text	id=error-message-text
19	priadrstaLocator	xpath=.//*[@id='content-sub']/form/div[1]/table/tbody/tr[7]/td[2]	xpath=//div[@id='content-sub']/div[2]/div/table/tbody/tr[4]/td[4]
20	secadrLocator	xpath=.//*[@id='content-sub']/form/div[1]/table/tbody/tr[8]/td[2]	xpath=//div[@id='content-sub']/div[2]/div/table/tbody/tr[5]/td[4]
21	wifiLocator	link=Wi-Fi	link=Wireless

Fig 3: Excel sheet database of web element Locators.

Fig 4: Read x-path value (accessLocator) from excel sheet to Hard code.

3.1.2 Architecture of proposed system.

The user input is a web page name. According to it the code will pair the web element locator values to the key values in the first column of the excel sheet. The web-driver drives the browser in which the web User Interface will run. The keyword execution results will be reported by the framework. The report will be in HTML format.

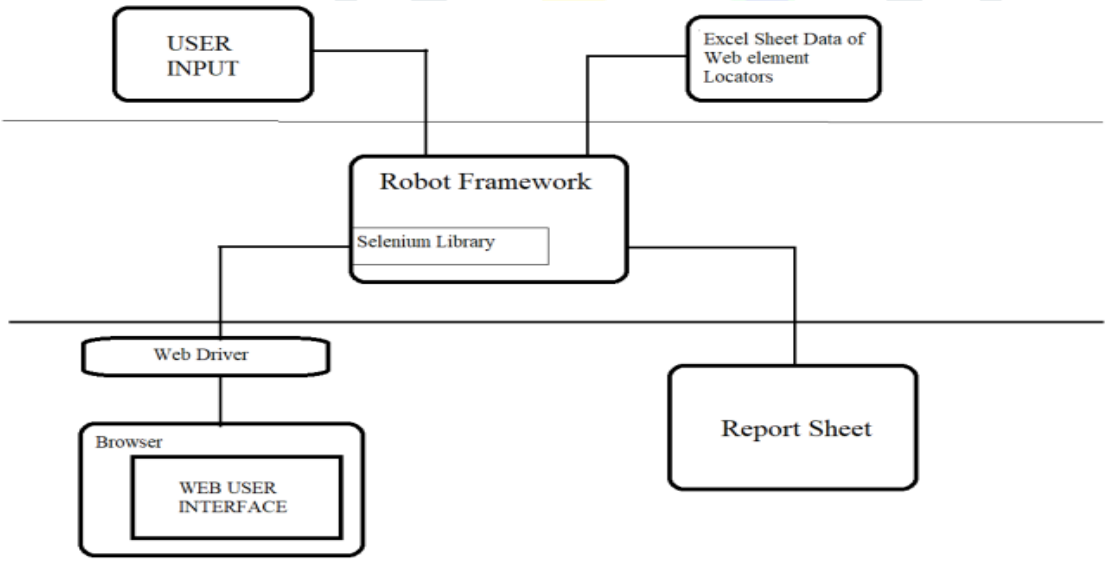


Fig 5: Architecture of proposed method.

3.1.3 Results

Using the proposed method, it is possible to make the test scripts constant and resistible to the web User interface field changes. If the new elements are added, we can add its element locator values like x-path to the excel sheet. Similar in case of element deletion from web UI also. The proposed method will reduce the effort of changing the test scripts every time when the Web UI changes by 70% – 80%.

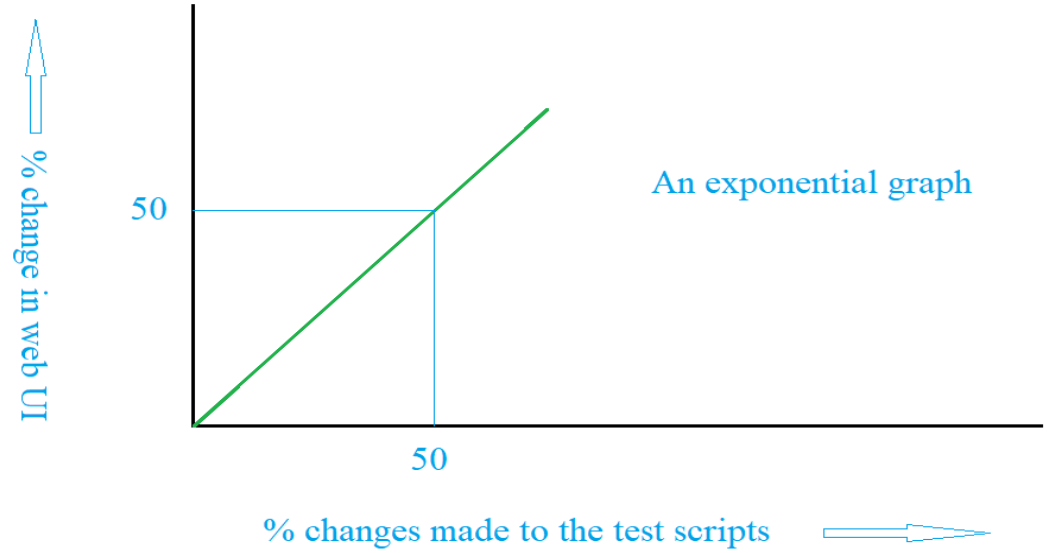


Fig 6: Graph for Normal test Scripting.

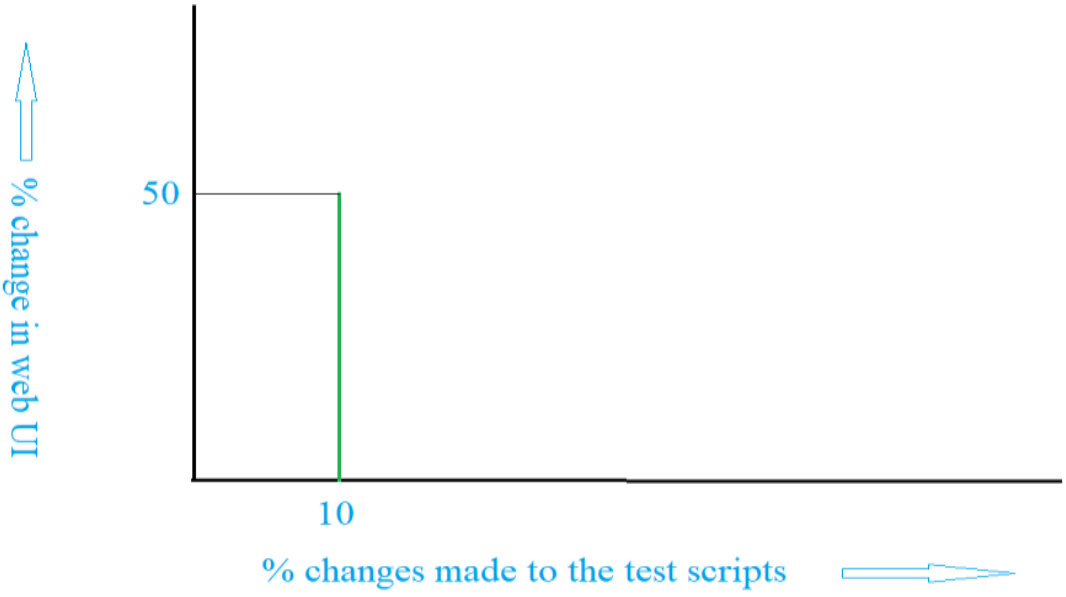


Fig 7: Graph for Proposed Approach.

3.2 An automated tool for generating report of the interactable fields in a web UI.

3.2.1 The Tool development

The developed tool will reduce the human effort of knowing the dynamic fields in a web page through which the user can interact and input the values to the dynamic fields of the web User Interface. For the tester it is important to know the various fields in a web UI and it types. The field types are important for both manual and automated testers for the type of values it will accept.

If the web page is large and have numerous fields to test. It is a time-consuming process in detecting each field which are interactable. The tool will automatically generate the report which will have a list of dynamic fields names, values of each field and the use of the fields.

The format of the excel sheet is as follows; the first column will have the name of the dynamic field; second column will have its type and the third column will have its use.

3.2.2 Algorithm Flow

The Source code of the web UI is taken as the input to the code. The code will then read the line and sorts them according to the input type values provided. The tool has leverage to fetch the new type of fields. These values are filled in the excel sheet to make the report readable. This report can be used by the testers to analyze the web user interface in many ways.

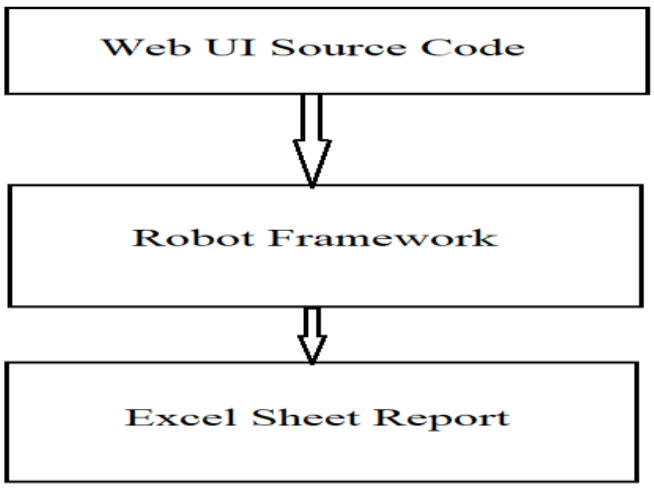


Fig 8: Architecture of automated report generator tool.

Sample web UI html code.

```
<tr>
<td></td>
<td><input type="checkbox" name="cbox" class="cbox" id="pforward" value="False"><b> Port Forwarding</b><br></td>
<td><input style="width: 400px; height:25px;" type="text" class="checkElements" id="ipforward" value=""></td>
<td>Default values</td>
</tr>
```

	A	B	C
1	**Fields**	**Type**	**Use**
2	Port Forwarding	Checkbox	Accepts either True or false value.
3	Save	button	onclick: submits the data values.
4	Reset	reset	resets the value.
5	User name	text	Accepts text filled in the Input box.

Fig 9. Excel sheet report.

3.2.3 Result

The report of the automated tool will considerably reduce the time in knowing the web page basic functionality, when a new web user interface arrives.

IV. Conclusion

The web user interface validation and testing are very important aspect. The bug in the interface will make the application incomplete and invalid. Hence, it is necessary to detect the flaws in the user interface before the release of the application. The use of proposed method will reduce the effort in changing the test scripts every time when user interface changes or updated. The use of automated tool that generates the report of the interactable fields in a web UI, will reduce the manual effort of knowing the interfaces which are user interactable. It is shown that the use of both the proposed method and automated tool will reduce the amount of manual work by 70 – 80 percent and reduces the time consumption by a margin of 50%.

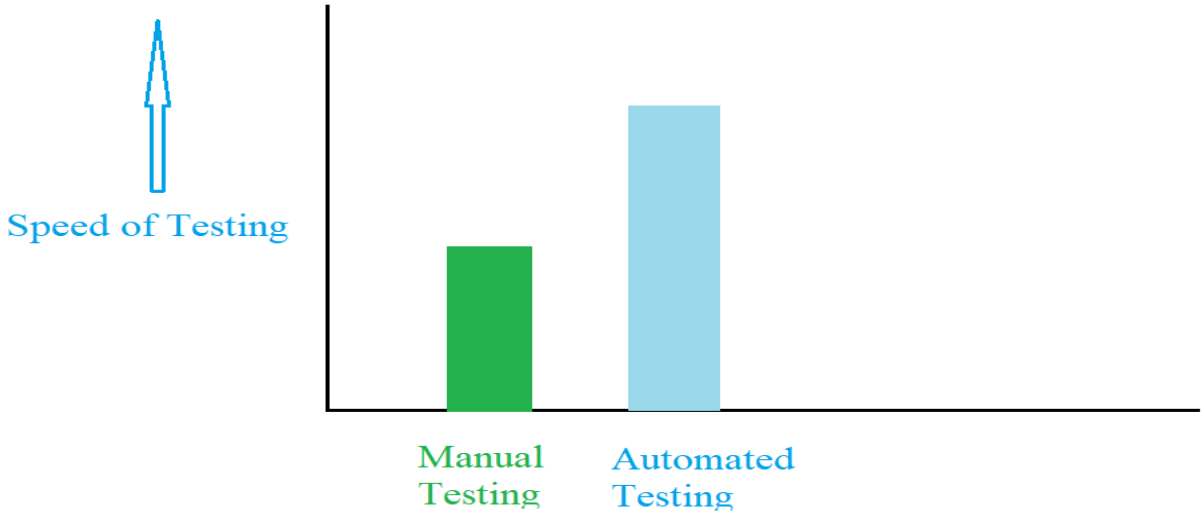


Fig 10: Speed of execution of Manual and automated testing.

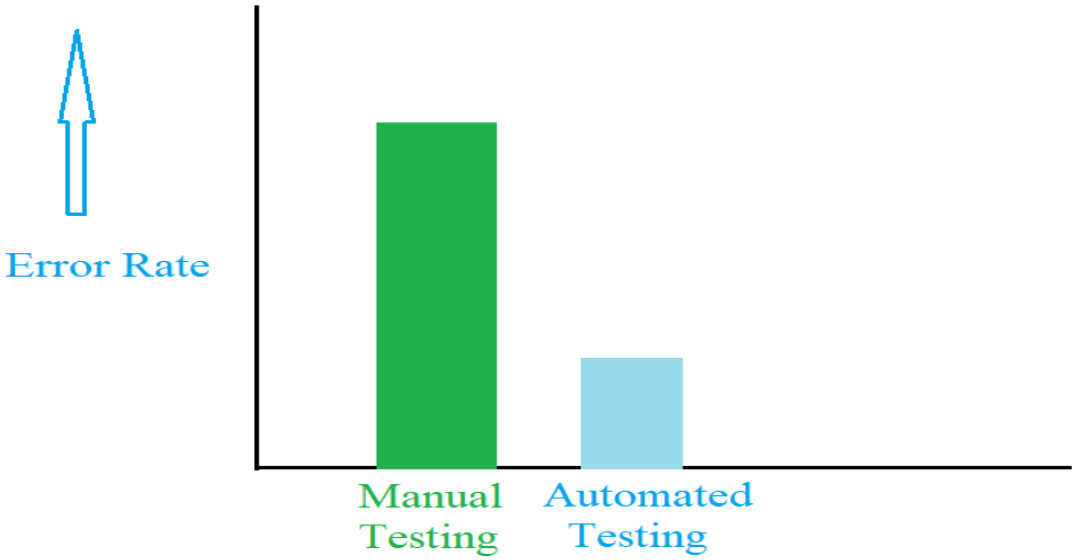


Fig 11: Error rates of Manual and Automated testing

REFERENCES

[1] Andreas Rau, Andreas Zeller, Jenny Hotzkow, "Poster: Efficient GUI Test Generation by Learning from Tests of Other Apps," ACM/IEEE 40th International Conference on Software Engineering: Companion Proceedings, 2018.

[2] Fei Song, Feng Xu, Zhuoming Xu, "An XPath-Based Approach to Reusing Test Scripts for Android Applications," 14th Web Information Systems and Applications Conference (WISA), 2017.

[3] Mika Katara, Tommi Takala, Tuomas Pajunen, "Model-Based Testing with a General Purpose Keyword-Driven Test Automation Framework," Fourth International Conference on Software Testing, Verification and Validation Workshops, 2011.

[4] Farzin Davari, Heidar Pirzadeh, Sara Shanian, "A Novel Framework for Creating User Interface Level Tests Resistant to Refactoring of Web Applications," 9th International Conference on the Quality of Information and Communications Technology, 2014.

[5] Robot Framework website, https://robotframework.org/ .