A TOOL BASED APPROACH FOR AUTOMATION OF GUI APPLICATIONS

P.Nagarani[1], R.VenkataRamanaChary[2]
[1]Padmasri Dr.B.V Raju Institute of Technology Hyderabad, India
nagarani.444@gmail.com
[2]Padmasri Dr.B.V Raju Institute of Technology Hyderabad, India
rvrchary@gmail.com

Abstract-**As the size and complexity of software is continually growing, manual testing becomes very difficult and tedious task. Also as the computer systems are significant to our society in every daily life and are performing an increasing number of critical tasks, so more work in software testing and analysis has become of great importance. The software applications once developed need to be maintained and tested as they undergo regular or frequent modifications. Automation of software testing and tool support for testing, therefore, has been emerging as a very important technology to quality assurance of present software industry.**

The proposed paper provides view of a new approach to automate the software testing process using Coded UI (User Interface) tool. Automation of the test plays a significant role in testing activity, as it saves time and provides better utilization of resources. The test automation has many situations to deal such as mapping of user specifications to identification of the test cases, test case generation, maintenance of test cases and test scripts.

Keywords: Test Automation, GUI Applications, New approach for Automation, Automation Methodology

1. INTRODUCTION

Currently software organizations are competitive in all the aspects, especially in terms of the quality of the products to be delivered and in parallel in the context of the time line also. The software complexity is increasing while releasing the high quality software within the time line is on raise. And in parallel, more and more defects are more probable due to the increase in the software complexity.

Automation of testing is an important aspect in software industry. It is only with automation that testing becomes practical and scalable to the size of a typical system with which the industry has to deal. Test automation can exploit not only knowledge from
the code under test but also from available models or specifications.

2.RELATED WORK

One solution for improving the effectiveness of software testing is to perform automation testing. In this approach, testers can focus on critical software features or more complex cases, leaving repetitive tasks to the test automation system.

It can have positive impacts in many areas, as the test automation will save money and solve some testing problems. Inadequate and ineffective testing is responsible for many problems regarding software reliability faced by computer users. On the other hand, the complexity of modern software packages makes manual testing difficult. But on the contrary, automated testing can help to improve efficiency of the testing process in order to identify areas of a program that are prone to failure. As personnel costs and time limitations are significant restraints of the testing processes, it also seems like a huge investment to develop test automation to get larger coverage with same or even smaller number of testing personnel.

The proposed Coded UI tool in VSTS (Visual Studio Team Server) is useful to create completely automated tests for the validation of the functionality and behavior of the application User Interface. Besides this many other commercial tools are present in the marketplace such as Quick Test Professional (QTP), Test Complete etc.

Apart from this many other open-source tools were also present. Test automation is a process of writing a computer program to do testing that would otherwise need to be done manually. Once tests have been automated, they can be run quickly and repeatedly. This is often the most cost effective method for software products that have a long maintenance life, because even minor patches over the lifetime of the application can cause features to break which were working at an earlier point in time.

Some of the Open source automation tools that were present were AutoIT (for Windows Application), Selenium (for Web-based applications), WatiR, WatiN etc.

The testing methods generally includes the Black-box testing and the White-Box testing. The white-box approach is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. Also here an internal perspective of the system, as well as programming skills, is used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. The black-box approach is the method where the software under test has to be verified with a suitable studied set of inputs whose expected outputs are known only on the basis of the functional specifications. The proposed framework can be considered to be the black-box approach.

There are two general approaches to test automation: a. Code-driven testing: The public interfaces to classes, modules, or libraries are tested with a variety of input arguments to validate that the results that are returned are correct. b .Graphical user interface testing: A testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct. Test automation tools can be expensive, and it is usually employed in combination with manual testing. It can be made cost-effective in the longer term, especially when used repeatedly in regression testing.

The general test method approach performed as: Initially the inputs are considered based up on the required functionality of the application to be tested. Then a test strategy is performed, i.e., the generation of the test cases based upon the valid inputs. The generated test cases are then executed and finally the results are obtained with the precise expected output. Based on the outcome of data, the Pass/Fail condition of test can be considered.

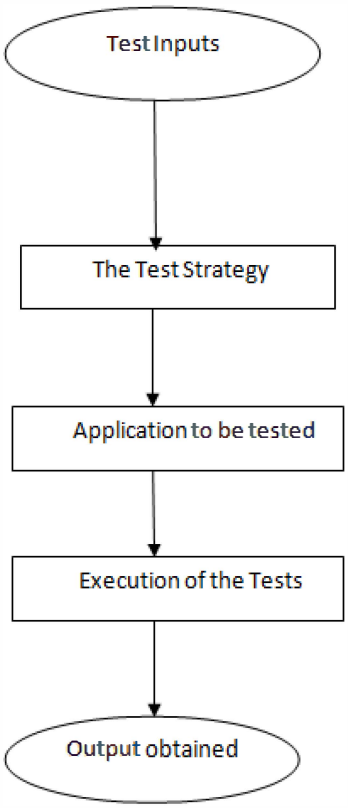

Fig.1.Block Diagram of General Testing Approach

1. METHODOLOGY OF AUTOMATION TESTING

Automation testing is a process of writing a computer program, i.e., a script to do testing that would otherwise need to be done manually. The scripts once written or developed for the application or the software under test can be run repeatedly as per the requirement. Also, it is a quick and efficient process without the manual intervention. As such the test coverage of the application, the maintenance of the scripts, un-attended modes of the user were observed to be beneficial in the automation testing process.

Automation Framework is not a tool to perform some specific task, but is an infrastructure that provides a complete solution where different tools work together in a unified manner hence providing a common platform to the automation engineer using them.

The Testing framework is responsible for

1. The general format to be understandable
2. A Framework to be created to handle the software application under test
3. Test execution
4. Reporting the results Scripting techniques are generally used:**1** .Linear (procedural code, possibly generated by tools like those that use record and playback) **2**. Structured (uses control structures - typically 'if-else', 'switch', 'for', 'while' conditions/ statements)

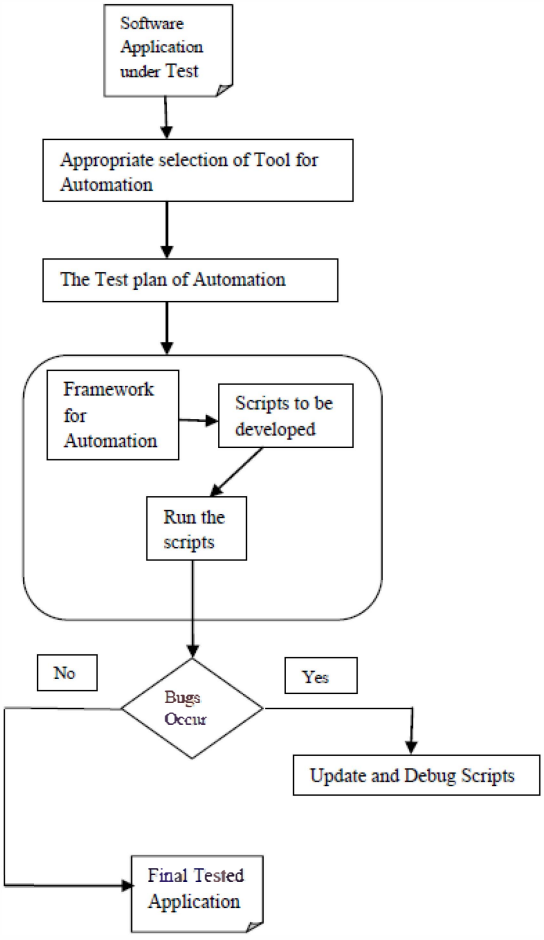

Fig.2.Process of Automation Testing

4. PROPOSED TOOL FOR AUTOMATION

The appropriate tool proposed for the Automation of the software application is Coded UI tool in VSTS 2010(Visual Studio Team Server). The Coded UI tool is useful to create completely automated tests for the validation of the functionality and behavior of the application User Interface. The coding standards of the tool were based up on the C# .net classes, methods, conventions etc.

Coded UI tool has the following capabilities:

1) Efficiency in Recording
2) Ability to generate scripts
3) Data Driven Testing
4) Reporting of test results
5) Script Reusability
6) Playback of the scripts

1) *Efficiency in Recording*

The Coded UI tool has the capability of Recording and playback of the applications of User Interface. It can capture all the keyboard strokes and mouse clicks that were performed on the application. All the controls in the application such as:
Textbox, Checkbox, Button, Radio Button,
Menu item, Context Menu, Mouse Clicks,
Keyboard strokes, Edit box, Combo box etc.

2) Ability to generate scripts

For all the recorded actions the tool has the ability to generate script accordingly and while running the script, steps are performed sequentially as how the actions are recorded.
All the script generated will be placed in a class file with the method names involving the functions, variables etc. in it.

3) Data Driven Testing

The Data Driven testing includes usage of the external files or the databases for the data to be imported into the script rather than hard coding the data inside the script. The external files include CSV (Comma Separated Values), XML (Xternal Markup Language) and also the database connection also.

4) Reporting of test results

The test results can be reported as the Passed/Failed statements. Also the results can be exported and reported in Xml files or the Excel files.

5) Script Reusability

The script can be reusable any number of times and so can be run based on the requirements. Also the execution speed will be very much higher.

6) Playback of the scripts

The playback of the scripts can be done many times after the recording of the user actions is performed.

4.1 Generic Framework Created For The Proposed Tool

By recording all the controls in the application can be captured and played back, i.e., the entire mouse clicks and the keyboard strokes can be recorded. But, on the contrary the generation of the script by the tool is huge and difficult for the user to understand. And so, we created a generic framework for each and every control so that it can be used for any application and also easily understandable. The controls/Objects in the application were identified by Name and the Control Name. Here the input can be taken from the Data table or can be hard coded in the script and the output can be reported in the Xml file/Excel file.

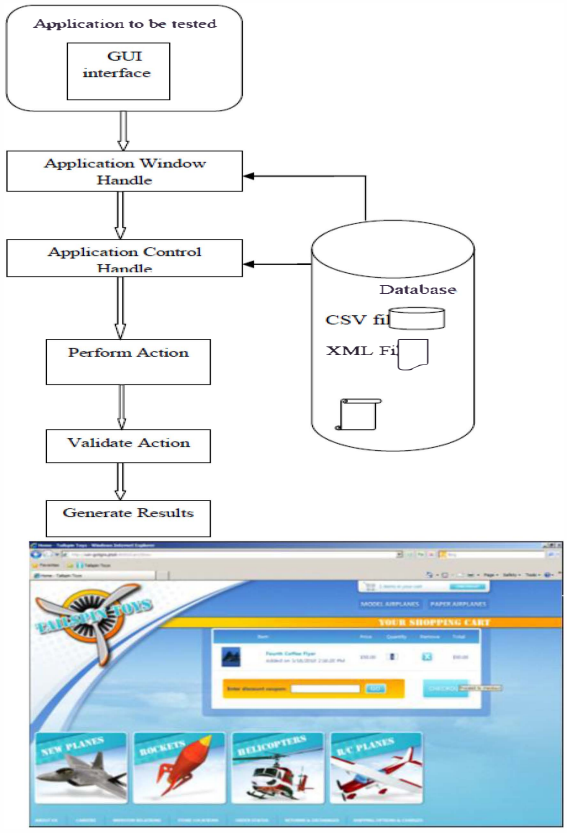

Fig.3.Generic Framework for Coded UI tool

the window name, then using unique control ID, the

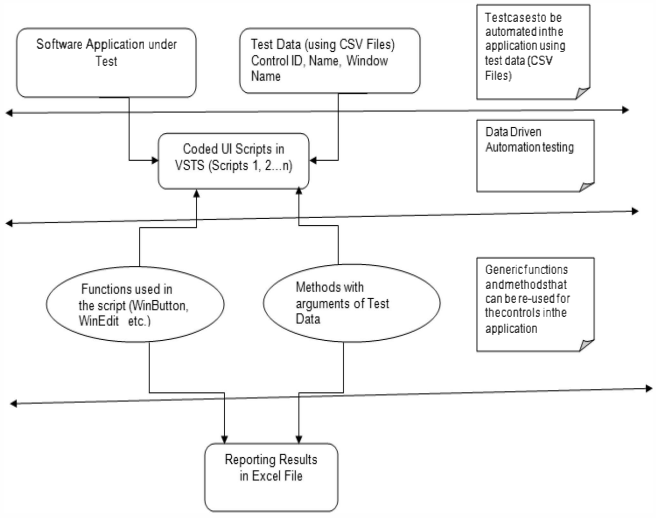

Fig.4. Architecture diagram for the generic framework

Fig.5. Sample Application

5. EXPERIMENTAL RESULTS

The Result can be observed in the following manner:

5.1 *Automation Metrics*

By performing the automated testing the overall test coverage is done and a huge amount of time is saved. The Time saved by automating an application can be given as: Difference (Manual hours spent for testing
the application) to the (Time spent in doing the automation of the application)

The automation metrics can be observed by using the following:
The Percentage of Automation

$$= \frac{\text{No. of Test Cases Automatable}}{\text{Total No. of Test Cases}}$$

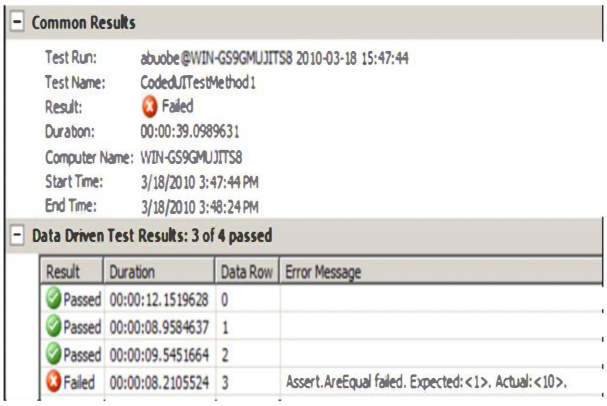

Fig.6.Result of the Application

Our application has almost 120 test cases involving a number of controls. When tested them manually, testing time is approximately 40 Hrs of execution time. But when automation testing is performed it is done using the scripts in an un-attended mode of the personnel and time is reduced to 2 Hrs of execution time.

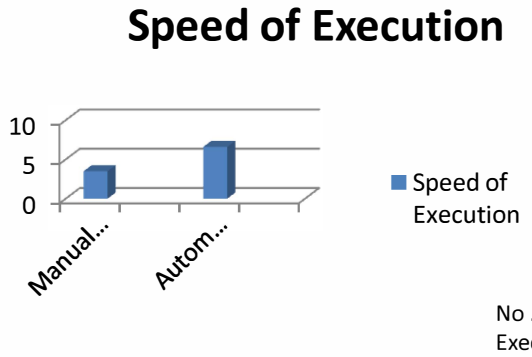

Fig.7.Speed of Execution of both Manual and Automation Testing

6. CONCLUSION

The paper proposes a new way for automation of the software applications involving complex functionalities and tasks. The proposed Coded UI tool and the generic framework created for the tool helps in automating the software applications under test with complete test coverage and saving the huge amount of time and resources.

As such each and every test case and the scenario can be automated with minimal time compared to the manual testing time.
Also the Automation using the tool helps in maintenance of the scripts in an understandable manner which can be reused any number of times depending on the requirement of the user.

7. REFERENCES

[1] R. Ramler and K. Wolfmaier, "Observations and lessons learned from automated testing," in Proceedings of the International Workshop on Automation of Software Testing

[2] K. Karhu, T. Repo, O. Taipale, and K. Smolander, "Empirical observations on software testing automation," in Proceedings
of the 2nd International Conference on Software Testing

[3] D. Galin, Software Quality Assurance: From Theory to Implementation, Pearson Addison Wesley, and Harlow, England.

[4] A. Birolini, Reliability Engineering — Theory and Practice

[5] O. Taipale and K. Smolander, "Improving software testing by observing causes, effects, and associations from practice," in Proceedings of the International Symposium on Empirical Software Engineering

[6] B. Shea, "Software testing gets new respect," InformationWeek

[7] M. Auguston, J. B.Michael, and M.-T. Shing, "Test automation and safety assessment in rapid systems prototyping," in
Proceedings of the 16th IEEE International Workshop on Rapid System Prototyping

[8] A. Cavarra, J. Davies, T. Jeron, L. Mournier, A. Hartman, and S. Olvovsky,
"Using UML for automatic test generation," in Proceedings of the International Symposium on Software Testing
and Analysis.

[9] M. Vieira, J. Leduc, R. Subramanyan, and J. Kazmeier, "Automation of GUI testing using a model-driven approach," in Proceedings of the International Workshop
mation of Software Testing

aochun, Z. Bo, L. Juefeng, and G. Qiu, "A test automation solution on gui functional test," in Proceedings of the 6th IEEE International Conference on Industrial Informatics

[11] D. Kreuer, "Applying test automation to type acceptance testing of telecom networks: a case study with customer participation," in Proceedings of the 14th IEEE International Conference on Automated Software Engineering

[12] Kerry, Automated Software Testing – A Perspective, Website testingstuff.com

[13] M.A. Bailey, T.E. Moyers, S. Ntafos, An application of random software testing, IEEE MILCOM, Conf. Rec.

[14] S.M. Phadke, Quality Engineering Using Robust Design, Prentice-Hall, Englewood Cliffs

[15] ANSI / IEEE Std. 1012, IEEE Standard for Software Verification and Validation Plans