Automated Software Test Optimization using Test Language Processing

Mukesh Mann[1], Om Prakash Sangwan[2], and Pradeep Tomar[3]
[1,3]Department of Computer Science and Engineering, Gautam Buddha University, India
[2]Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology, India

Abstract: *The delivery of error free software has become a major challenge for software practitioner since many past years. In order to deliver an error free software testers spends 40-50 % software design life cycle cost during testing, which further get incremented with changing user demands. The Large existence of test cases for a particular functionality is possible and some of them may cause software fails. Thus it raises a demand to automate existing approach of manual testing which can minimize execution efforts while maintaining the quality of testing. In this paper, a regression framework based on keyword oriented data-driven approach has been proposed for generation and execution of test cases. The methodology for the developed framework is based on Test Language Processing (TLP) which acts as a comprehensive approach to design and execution of test cases. The framework is tested on an open source web application called Vtiger-Customer Relationship Management (CRM) version 5. The framework is compared against manual testing in terms of test suite execution and their optimization. Based on our experiments it is concluded that (1) Test execution time using TLP based framework is significantly low and (2) a test suite optimization of 83.78% is achieved through the proposed TLP framework.*

Keywords: *(TLP) framework, manual testing, effort reduction, test optimization.*

Received June 17, 2015; accepted July 4, 2016

1. Introduction

Recent years have seen a dramatic growth in the software industries whose focus is on fast delivery of client's requirements. In order to maintain rapidness in delivery, functional quality of software system has become a new subject of interest. In order to meet this demand various software testing techniques have been proposed, for instance, Boundary Value Analysis (BVA) in which test cases are designed from a given input domain whose values are either on the boundary or near to boundary.

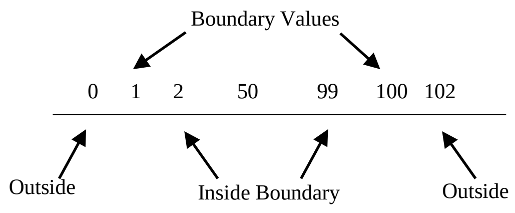


Figure 1. Input values for a program.

Such test cases results in high fault detection [12]. Extension to BVA is robustness testing through which invalid test cases are selected, and observe the program under test. Two more states are induced, one below and other above the minimum and maximum value respectively, i.e., 0 and 102 as shown in Figure 1. Other frequently used testing techniques include control flow, coverage [5, 26, 27, 28, 31, 36] path, data flow, and loop testing [23, 37].

Many similar approaches are there to test a software system with one common goal which is to decrease test time and increase faulty coverage. Thus to apply and implement similar testing techniques, a traditional manual approach is usually followed in most practical scenarios in which designed test cases are executed by manual tester. Certainly, such techniques save huge time in designing effective test cases but the efforts consumed in design and manual execution of these designed test cases using prior techniques are very high. Therefore the requirement to establish a test automation framework in organization progressing towards mature quality assurance model has become a demand in today's agile environment. Automation tools are used to design and execute effective test cases, therefore investment in such tools is a subject of research and it is always stated that careful investments in any such tools may decide the success and failure of an enterprise [18, 34, 36, 41].

In this paper, a regression framework is designed and developed for functional test cases and to compute the level of test case optimization achieved during regression cycle in comparison to manual testing.

Section 2, gives introduction about the related work in this area. Section 3 discusses in detail the methodology and framework. Section 4 discusses the experimental setup and design. In section 5 performance evaluation of the proposed framework is done. The effort reduction using proposed framework is discussed in section 6 and finally, the paper is summarized by discussing conclusion and future prospects in section 7.

2. Related Work

Previous studies in software engineering estimated that more than fifty percent of development cost is consumed in software testing, and some studies also pointed out the high economic impact on the United States due to poor testing infrastructure [9, 25]. Therefore the need for improving the existing testing infrastructure and development of better software testing techniques that can meet up the demand for today's complex software has opened up the door for further research in the areas such as the design of effective test cases that can validate dynamic user requirements and how to execute them in minimum time. These new challenges go further to open up a debate over longstanding problems such as how to quantify and evaluate more robust testing criteria and how to minimize regression testing efforts.

In last few years, automated software testing has given a huge focus in the software industry as verified by many specific events, conferences, and workshops across the globe. But the two main important aspects of software testing i.e., test data generation and their execution is still in infancy.

Although many researchers have proposed many techniques [6, 8, 14, 15, 29, 36] for automatic test case generation which has overwhelm reduced the burden of manually writing unit test cases but still we are missing a general purpose tool which can test a software system with great zeal.

One possible method to improve the yield of automation testing is to use the method of formal specifications that can guide test data generation and execution [12, 17, 24] but unfortunately, most of these specifications are missing in practice. A number of automation tools for testing are available such as Quick Test Professional-(QTP) (Mercury), Rational tool from IBM and Selenium as an open source. The wonderful advantage of these tools is in automatic execution of manually created test sequences without the need for manual intervention. Among many available testing tools QTP [24] and Selenium are used by most software testing practitioners. The use of a particular tool depends on a number of factors such as cost, availability, proficiency, easiness and the scripting time. A few advantages of considering QTP for testing solution by many organizations are summarized below

- The language used is Visual Basic (VB) script which is very easy to learn and the organization does not really require skilled coders to work with it.
- The object repository is a great feature in this tool with which the team can meet up the demand for today's component-based orientation and web service testing.
- An excellent technical support which is missing in selenium due to its open source nature and hence developers have to rely on community support.

Although our motivation is not to underestimate Selenium as it has many other features that arguments it applicability but again such choices have always been biased by the organizations testing requirements. Our motivation in this paper is to propose a regression framework through TLP methodology using QTP, But before that a brief of available Regression Test case Selection Technique (called as RTS) is necessary, with various RTS techniques proposed in literature one can select test case's from test suite (T), such a T validate if any previously modified part of software is continuously working without causing any error condition. Thus RTS has an advantage in reducing efforts such as testing cost when working in the dynamic environment. Table 1, briefly summarizes the comparison of such techniques [4].

Table 1. Comparison of various RTS techniques.

RTS Technique	Contributor(s)	Criteria	Pros	Cons
Data Flow Analysis	[3, 22, 23, 33]	Data flow in programs and it structure	Modifications like intraprocedural and interprocedural are easy to analyze.	Low safety level and highly imprecise
Slicing Techniques	[1. 5]	Involves slicing in programs and dependency graph models	Can analyze intraprocedural and interprocedural modifications.	Imprecise and low-level safety. It also involves high cost as compared to other data- flow methods
Module Based Firewall Techniques (MFT)	[14, 19, 38]	Modules dependencies are analyzed	Due to analysis of source code of modified modules, it is more efficient than other regression techniques	Low safety and high imprecision level
Modified Code Entity Technique	[42]	High granularity level can be adapted	High efficiency and safety make it most safe RTS among other techniques.	imprecise nature
Textual Based Differencing Techniques (TBDT)	[9, 8, 10, 16, 32]	Textual based differencing of procedural programs like c	Easy to implement and average safety level.	Safety is average and it is difficult to adjust TBDT to current languages. Also for big programs, its efficiency is too low
Graph Walk Technique (GWT)	[12]	Flow graph are analyzed in depth	Safe and precise in nature	Efficiency is less than [12, 14]
Database Techniques	[7, 32]	All states in database need to be taken into consideration	Safety is high.	Precision is low
Web Based Techniques	[2, 7, 11, 20, 21, 40]	Source code for web service cannot be used for analysis of whole web page..	This technique is safer and system designed using [2] is highly efficient in comparison to techniques proposed by [11, 21]	The precision level depends on the net content of information in a module, thus a varying level of precession can be observed with different modules.
AspectJ Techniques	[13, 16, 39]	Dependencies occurrence due to join-points and Pointcuts must be taken into account	The technique proposed by [13] is more safe as compared to [16].	These techniques are computationally high in cost.

In the next section, the methodology and the framework are introduced to:

1. Automate the process of manually executing test cases.
2. To optimize functional test cases so as to minimize regression testing efforts.

3. Methodology and Framework

- Definition: Test Language Processing (TLP): “A tester-specified dictionary of keywords and parameters that facilitate communication among testers and other subject-matter specialists” [22].

Where the keyword is an English word specified by the manual tester as per their level of understanding. Usually, it is recommended to specify meaningful keyword so that it becomes easy to understand the context in which they were used.

The TLP can be seen as a dictionary of keyword and their parameters. Various elements present over any WebPages/windows application are called an object for that application. Form fields such as user_name and password are an example of an object. An object itself constitutes a dictionary. While testing any such system the dictionary plays an important role. A logical value to an object is called its parameter and whatever the operation(s) carried out on such an object represents its keyword.

For Example: consider a web page that has constituted a login screen having:

1. user_name.
2. User_password fields.

Such fields are termed as objects and their logical names are “user_name” and “user_password”. The type of these objects is “INPUT” which means a user is allowed to enter text in these objects. Therefore the “INPUT” is termed as a keyword to these objects. The value entered by the user in the object is called its parameter. In summary, an object constitutes an object’s logical name and its dictionary (keyword and parameter) as shown in Figure 2.

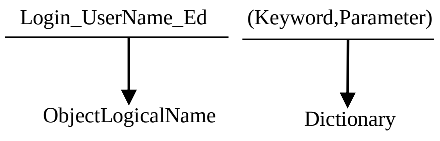


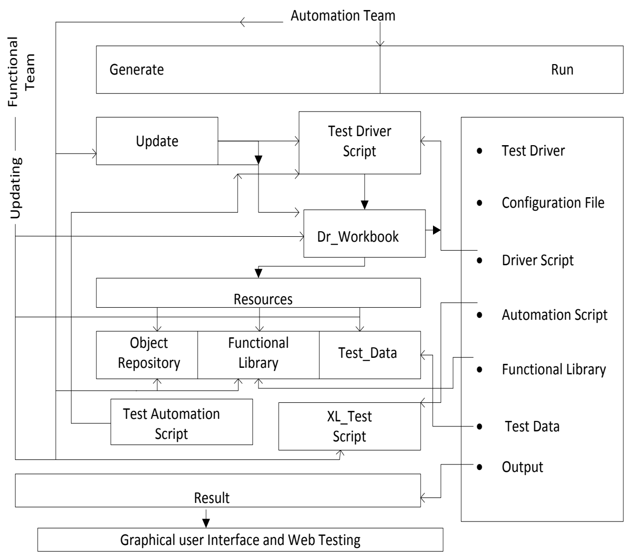
Figure 2. The structure of TLP.

The two teams play a vital role in TLP infrastructure management are:

1. Functional testing team.
2. Automation testing team.

Functional testing is responsible for adding and updating new keywords whereas the automation team’s responsibility is to design, develop and manage test scripts on the basis of keywords supplied by the functional team.

A complete regression framework using TLP methodology for reducing the manual testing time and for achieving the highest level of test optimization during regression testing is shown in Figure 3.



Web or GUL Application Testing

Figure 3. The Framework.

A brief overview of various terms appeared in proposed methodology are discussed below.

- Driver-the driver script and its workbook are contained in a folder called Driver.
- Test Driver Script-The code for the framework is implemented in a workspace called Test driver script.
- Driver workbook-an excel sheet that contains the sequence of tasks/ modules to be tested.
- Test Automation Script-Actual keywords as specified by the functional tester are contained in an excel sheet called XL_test_script which is further put in a folder named Test Automation Script. The sequence which is specified in XL_test_ scripts are derived by the Driver Workbook.
- Functional libraries-various functions are implemented for various requirements and keywords. These functions are well coded in a workspace. The name of such workspace is called Functional libraries.
- Object repository-the various objects in an application is identified through the object repository that contains number of predefined identifiers to identify a particular object.
- Test data: the data to be supplied while testing an Object. It is specified in an excel sheet called test data.
- Output-this folder holds the testing report in various formats such as excel, image, text format.

Thus the TLP infrastructure has two main teams as

a) *Functional testing team*-specify which module (s) is /are to be tested for a given application and timely update keywords sequence.
b) *Automation testing team*-implements the keywords sequence in the form of test scripts.

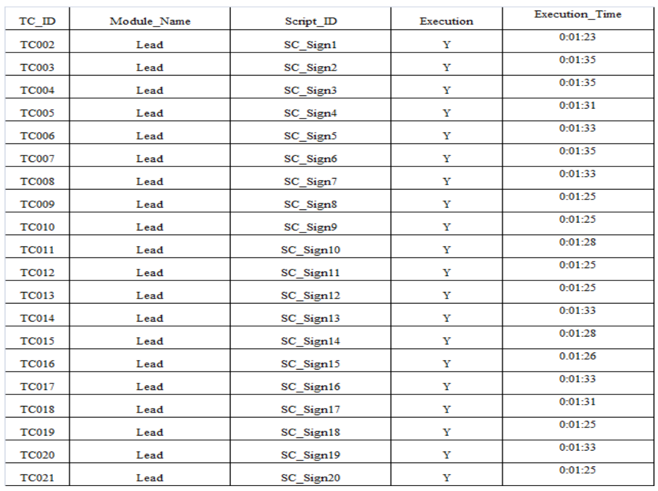
The collaborative efforts of both teams result in effective framework development. Following are the main steps for design, development, and execution of framework.

- The test sequences specified by the functional team in XL_test_script are implemented by the automation team by writing the code using VB script. Before that, loading of the configuration file is done to specify the path of SUT.
- The automation team implements the driver script in such as way that it calls the automation script as per the sequence of driver script. The sequence specified in the automation script cannot be implemented without the help of functional libraries. Thus the selection of functional libraries according to the requirements is very necessary for the success of automation process.
- After the development of the framework, the execution is carried out by the automation team. The update of various elements is done by both teams in collaboration as shown in methodology
- The output is stored in excel file that contains detailed information about the result such as total testing time for individual modules, a number of passed/failed test cases and snapshots of failed test cases.

4. Experimental Setup and Design

The performance of proposed framework using the TLP methodology is validated using an open- source application [35]. The large size and high complexity of this Customer Relationship Management (CRM) application ensure more possibility for testing modules in depth and it also builds confidence in the developed framework for more complex and large size applications. The SignIn-Signout module of this application is chosen to test the framework. Also in a later phase, a regression testing framework is also developed to check application by deleting many existing functionalities including various forms, buttons, and Text name. This is done in order to check the working of the framework under heavy changes in application and to make sure that the framework is able to capture errors during regression testing with great zeal.

The SigIn-Signout functionality of Vtiger application is considered as a Module Under Test (MUT), for which 20 subjects or functional testers each having an industrial experience in testing in a range of 1-3 years were selected to functionally test the MUT with dataset [22] obtained using (BVA). The average testing time for the MUT is obtained by asking each subject to test it twenty times using the given dataset. In this way, we get manual test readings for the MUT. The same work is the carried out using the developed framework by executing it on QTP 10.00 [30] and windows 7 version having the configuration of 64 bit and 4GB RAM. The Driver script for MUT as made by the automation tester is first called by the executed framework. A complete driver script for the MUT is shown in Figure 4.



TC_ID	Module_Name	Script_ID	Execution	Execution_Time
TC002	Lead	SC_Sign1	Y	0:01:23
TC003	Lead	SC_Sign2	Y	0:01:35
TC004	Lead	SC_Sign3	Y	0:01:35
TC005	Lead	SC_Sign4	Y	0:01:31
TC006	Lead	SC_Sign5	Y	0:01:33
TC007	Lead	SC_Sign6	Y	0:01:35
TC008	Lead	SC_Sign7	Y	0:01:33
TC009	Lead	SC_Sign8	Y	0:01:25
TC010	Lead	SC_Sign9	Y	0:01:25
TC011	Lead	SC_Sign10	Y	0:01:28
TC012	Lead	SC_Sign11	Y	0:01:25
TC013	Lead	SC_Sign12	Y	0:01:25
TC014	Lead	SC_Sign13	Y	0:01:33
TC015	Lead	SC_Sign14	Y	0:01:28
TC016	Lead	SC_Sign15	Y	0.01:26
TC017	Lead	SC_Sign16	Y	0:01:33
TC018	Lead	SC_Sign17	Y	0:01:31
TC019	Lead	SC_Sign18	Y	0:01:25
TC020	Lead	SC_Sign19	Y	0:01:33
TC021	Lead	SC_Sign20	Y	0:01:25

Figure 4. Driver_script for MUT.

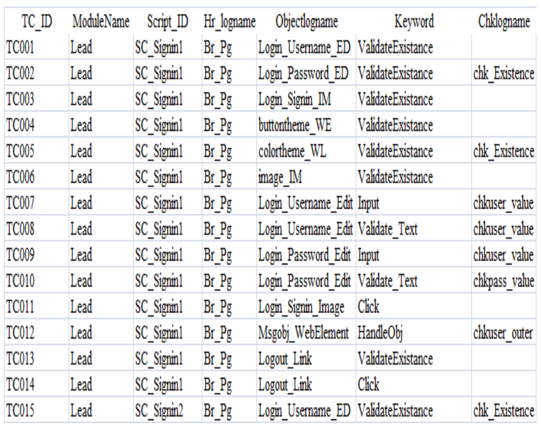
The key elements of Driver Script are:

- Tc_ID-randomly chosen test case id is assigned with each Script_id. This helps to identify functionalities which need to be tested.
- Module_name- The name of MUT to be tested, here “Lead” is the MUT in driver_script.
- Script_id-It represents the tasks to be performed. For example “SC_SIGNIN_1’’ in Script_id represents that user had login for the first time. For the second time, the script id is SC_SIGNIN_2. The automation script runs according to these sequences.
- Execution-It represents which Script_id the tester wants to run. The symbol “Y” (yes) in front of any script-id represents that tester want to run that id whereas the symbol “N” stands for No.
- Execution_time- It counts the total execution time for a particular Script id.

Automation tester develops scripts as per the sequences specified in Script_id column of Figure 4.
The automation script is shown in Figure 5.
The key elements of Automation script are:

- TC ID-It is used to track which test scripts were failed/passed during the execution.
- Module name-the module to be tested.
- Script id-performs the same task as performed in driver_script. More the functional requirements more will be the steps required to execute a script_id.

- Object logical name-It is the logical name of an object as specified by the tester.
- Keyword-the operation to be performed on a particular object is identified by the keyword. For instance, the keyword “CLICK” represents a mouse click operation on an object in the application.
- Chklogname-the validation of supplied input and/or the properties of a current object is verified using Chklogname filed.



TC_ID	ModuleName	Script_ID	Hr_logname	Objectlogname	Keyword	Chklogname
TC001	Lead	SC_Signin1	Br_Pg	Login_Username_ED	ValidateExistance	
TC002	Lead	SC_Signin1	Br_Pg	Login_Password_ED	ValidateExistance	chk_Existence
TC003	Lead	SC_Signin1	Br_Pg	Login_Signin_IM	ValidateExistance	
TC004	Lead	SC_Signin1	Br_Pg	buttontheme_WE	ValidateExistance	
TC005	Lead	SC_Signin1	Br_Pg	colortheme_WL	ValidateExistance	chk_Existence
TC006	Lead	SC_Signin1	Br_Pg	image_IM	ValidateExistance	
TC007	Lead	SC_Signin1	Br_Pg	Login_Username_Edit	Input	chkuser_value
TC008	Lead	SC_Signin1	Br_Pg	Login_Username_Edit	Validate_Text	chkuser_value
TC009	Lead	SC_Signin1	Br_Pg	Login_Password_Edit	Input	chkuser_value
TC010	Lead	SC_Signin1	Br_Pg	Login_Password_Edit	Validate_Text	chkpass_value
TC011	Lead	SC_Signin1	Br_Pg	Login_Signin_Image	Click	
TC012	Lead	SC_Signin1	Br_Pg	Msgobj_WebElement	HandleObj	chkuser_outer
TC013	Lead	SC_Signin1	Br_Pg	Logout_Link	ValidateExistance	
TC014	Lead	SC_Signin1	Br_Pg	Logout_Link	Click	
TC015	Lead	SC_Signin2	Br_Pg	Login_Username_ED	ValidateExistance	chk_Existence

Figure 5. Test script for automation.

Finally, the framework is executed and results summary is shown in Table.

Table 2. Result summary.

Pass	Failed	Total						
122	199	321						
Tcid	Modulename	Scriptname	Objectlogname	Status	Screenshots	Prpname	Expval	Actval
TC001	Lead	SC_SIGNIN1	Login_Username_ED	Pass				TRUE
TC002	Lead	SC_SIGNIN1	Login_Password_ED	Pass				TRUE
TC003	Lead	SC_SIGNIN1	Login_Signin_IM	Pass				TRUE
TC004	Lead	SC_SIGNIN1	colortheme_WE	Pass				TRUE
TC005	Lead	SC_SIGNIN1	logintheme_WL	Pass				TRUE
TC006	Lead	SC_SIGNIN1	loginimage_IM	Pass				TRUE
TC008	Lead	SC_SIGNIN1	Login_Username_ED	Pass		value	admin	admin
TC010	Lead	SC_SIGNIN1	Login_Password_ED	Fail	Snapshot	value	admin	admin1
TC012	Lead	SC_SIGNIN1	Loginmsgobj_WE	Fail	Snapshot	outertext	admin	*U&P
TC012	Lead	SC_SIGNIN1	Loginmsgobj_WE	Fail	Snapshot	outertext	admin	*U&P
TC012	Lead	SC_SIGNIN1	Loginmsgobj_WE	Fail	Snapshot	outertext	admin	*U&P
TC012	Lead	SC_SIGNIN1	Loginmsgobj_WE	Fail	Snapshot	outertext	admin	*U&P
TC012	Lead	SC_SIGNIN1	Loginmsgobj_WE	Fail	Snapshot	outertext	admin	*U&P
TC012	Lead	SC_SIGNIN1	Loginmsgobj_WE	Fail	Snapshot	outertext	admin	*U&P
TC012	Lead	SC_SIGNIN1	Loginmsgobj_WE	Fail	Snapshot	outertext	admin	*U&P
TC012	Lead	SC_SIGNIN1	Loginmsgobj_WE	Fail	Snapshot	outertext	admin	*U&P
*Indicates that you must specify a valid username and password								

The result summary is briefly discussed below.

- Propname-Objects in the application is validated against their property values. The Propname indicates its property value. If the required property (Chklogname) as mentioned in the driver script is not matched with the Propname then a test gets a failed status.
- Actval-The actual value achieved during the actual execution of the framework.
- Screenshot- It represents the snapshots for the failed test cases.
- Expval-it represents the expected outcome according to the client’s requirements.

All other Labels i.e., Module, Tci_id, Script_name and ObjectLogicalName are same as discussed before

Table 3 indicates average testing time in manual approach.

Table 3. Time in manual approach.

Iteration_count	No. of Test_case executed	No. of Passed Test _case	No. of Test _case Failed	Total_Time
1	19	10	9	1200
2	13	6	7	1440
3	13	4	9	1200
4	13	4	9	1260
5	13	4	9	1020
6	13	4	9	1320
7	13	4	9	1200
8	13	4	9	1380
9	13	4	9	1500
10	13	4	9	1110
11	13	4	9	1260
12	13	4	9	1380
13	13	4	9	1320
14	13	4	9	1440
15	13	4	9	1260
16	13	4	9	1500
17	13	5	8	1380
18	13	4	9	1440
19	13	4	9	1320
20	13	4	9	1260

The testing time using proposed TLP based framework is shown in Table 4.

Table 4. Time in TLP based approach.

Iteration_count	No. of Test_case executed	No. of Passed Test _case	No. of Test _case Failed	Total_Time
1	17	8	9	75
2	16	6	10	80
3	16	6	10	80
4	16	6	10	80
5	16	8	8	80
6	16	6	10	80
7	16	6	10	80
8	16	8	8	76
9	16	6	10	74
10	16	6	10	76
11	16	6	10	76
12	16	8	8	76
13	16	6	10	76
13	16	6	10	75
15	16	8	8	77
16	16	6	10	75
17	16	6	10	78
18	16	8	8	76
19	16	6	10	75
20	16	6	10	77

For performance analysis of the methods, a correlation and regression analysis is performed

5. Performance Analysis

The regression analysis [10] for each approach is shown in Figures 6 and 7.

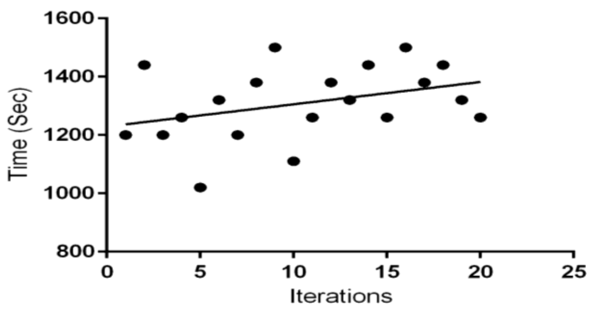


Figure 6. Regression analysis in manual testing approach.

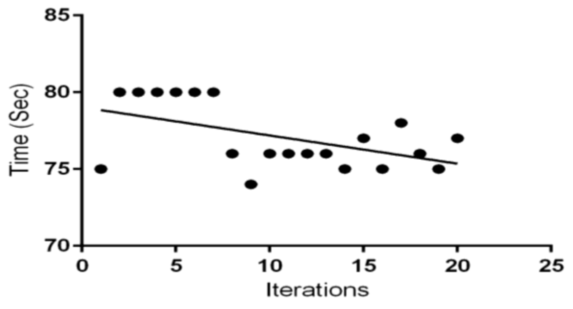


Figure 7. Regression analysis in TLP based testing approach.

The standard error obtained between the regression line and various data points are found to be 0.260 for manual approach and 0.126 for the proposed automation approach. A final comparison between both the methodologies is shown in Figure 8.

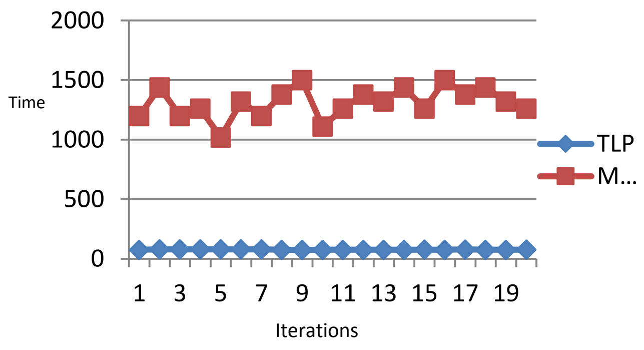


Figure 8. Manual vs. TLP efficiency.

Thus, the time remains almost same with successive iterations in proposed framework i.e. time taken to reveal a fault during regression cycle is same. While in manual testing, it depends on the speeds, accuracy, and performance of manual tester.

6. Effort Reduction using Regression Framework

With regression testing the tester re-test all modified parts of the software and ensures that any change to the software has not introduced any error in previously tested functionalities and also ensures that the newly added functionalities are working fine. With TLP methodology we have also developed a regression testing framework by extending the SigIn-Signout framework. The developed regression framework is used for retesting some old functionalities of a CRM-5 Module-Create_Lead. This module has a number of forms, Text fields, Buttons, and text area. We first check each area manually and then some of the functionalities of this module are removed. Our motive is to check how much time a tester will take to identify these changes manually and to see whether the developed regression framework is more efficient in finding the changes than manual approach.
Given below are the steps followed to measure regression time for Create_Lead Module of Vtiger application

1. Open the Vtiger's Create_Lead Module in local host.
2. Some functionalities of module Create_Lead are removed as shown in Table 5.
3. The developed regression framework for Create_Lead is launched in QTP.
4. Execute the framework and note total regression testing time along with a count of passed/failed test cases.

The testing time obtained using regression framework is compared against manual testing time. Our basic assumption during regression testing is that any change to the application will affect previous functionalities. With this assumption, some functionality are intentionally removed from the module Create_Lead as shown in Table 5. The developed regression framework is then executed on the Create_Lead with removed functionalities and the regression testing time using developed framework (Table 6) is compared against manual regression testing time. As manual tester has to re-test all functionalities of the Create_Lead module by re-executing all previous test cases, therefore the manual testing time will remain same as it was before removing the functionalities.

Table 5. Removed functionalities from create_lead module.

S.no	functionality	Removal Status
1	Lead_Fax	Y
2	Lead_Email	Y
3	Lead_Phone	Y
4	Lead_Mobile	Y
5	Lead_firstname	Y
6	Lead_Leadsource	Y
7	Lead_Leadstatu_WL	Y
8	Lead_No	N
9	Lead_Lat Name	N
10	Lead_Company	N
11	Lead_Title	N
*Y-Yes| N-No		

Table 6. TLP execution time.

TC_ID	Module_Name	Script_ID	Execution	Execution_Time
TC01	Lead	SC_SIGNIN1	Y	0:01:19
TC02	Lead	SC_CreateLead	Y	0:01:05
TC03	Lead	SC_EditLead	Y	0:00:09
TC04	Lead	SC_SaveLead	N	0:00:08
TC05	Lead	SC_Delete	Y	0:00:05
TC06	Lead	SC_Logout	Y	0:00:03

Both manual and TLP based regression framework approach are compared in Table 7. It is clear that with manual regression testing it takes 37 test cases to completely re-check all functionalities in total 300 sec while in TLP based regression framework only 6 test cases in 161 sec are sufficient to do so.

Table 7. Total testing time in manual vs. regression framework.

S.no	Manual	Regression framework
Total_ test_ case	37	6
Total_Time (in sec)	300	169

Therefore, the percentage reduction in test cases is calculated using following formula % test case reduction=(total test cases in manual-total test case in regression framework)/ (total test cases in manual) *100.Thus, % reduction=((37-6)/37)x100=83.78%

In this way, a high level of test case reduction is achieved using the TLP based framework.

7. Conclusions and Future Prospects

In this paper, a Regression Framework (TLP based) is developed for test suite execution and optimization in minimum time. With detailed analysis, it has been observed that in manual testing the time value depends on each successive iteration because in manual approach the testing depends on the human efficiency which itself depends on a number of factors such as-experience and tester's domain knowledge. But with the developed framework it has been observed that successive cycles of iterations do not play a much vital role as far as time is concerned. With each iteration, the testing time remains same for MUT. Thus this framework helps in carrying out testing at a rapid rate.

The framework is further extended to check if it is able to reduce manual regression efforts. With the successful execution of regression framework, it has been verified that 83.7 % of test suite reduction is achieved along with fully tested MUT which is not possible through the mannual approach in time bound environment. The proposed solution is not only applicable for web applications but also to mobile and desktop applications testing. In starting phase of paper the SignIn-Signout module was tested to build confidence in the proposed system. But in later phase i.e., during regression framework development, the proposed solution is tested on the larger module (Create_Lead) that contains a number of forms, input texts and radio buttons. The smooth working of framework biased its application to more complex and larger systems.

On problem faced during the development of proposed solution is the time taken to develop requirement oriented test scripts. The time can be minimized by giving proper training to the automation team. It is also pointed out that script development time becomes less important when deploying the large scale complex application.

In future, this work can be extended to check whether more optimized test scripts are possible to decrease script development time and also managing whole testing infrastructure for more complex benchmarks under a single solution.

Acknowledgement

This research work is supported by University Grant Commission, Govt. of India under Grant No. F/ NFO201415OBCDEL16123

References

[1] Bates S. and Horwitz S., "Incremental Program Testing Using Program Dependence Graphs," *in Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages*, Charleston, pp. 384-396,1993.

[2] Bicevskis J., Borzovs J., Straujums U., Zarins A., and Miller E., "SMOTL-A System to Construct Samples for Data Processing Program Debugging," *IEEE Transactions on Software Engineering*, vol. 5, no. 1, pp. 60, 1979.

[3] Binkley D., "Semantics Guided Regression Test Cost Reduction," *IEEE Transactions on Software Engineering*, vol. 23, no. 8, pp. 498-516, 1997.

[4] Biswas S., Mall R., Satpathy M., and Sukumaran S., "Regression Test Selection Techniques: A Survey," *Informatica*, vol. 35, no. 3, pp. 289-321, 2011.

[5] Boujarwah A. and Saleh K., "Compiler Test Case Generation Methods: A Survey and Assessment," *Information and Software Technology*, vol. 39, no. 9, pp. 617-625, 1997.

[6] Boyer R., Elspas B., and Levitt K., "SELECT-A Formal System for Testing and Debugging Programs by Symbolic Execution," *ACM SIGPlan Notices*, vol. 10, no. 6, pp. 234-245, 1975.

[7] Chaudhary N., Sangwan O., and Arora R., "Event-Coverage and Weight Based Method for Test Suite Prioritization," *International Journal of Information Technology and Computer Science*, vol. 6, no. 12, pp. 61-66, 2014.

[8] Chen Y., Rosenblum D., and Vo K., "TestTube: A System For Selective Regression Testing," *in Proceedings of the 16th International Conference*

on Software Engineering, Sorrento, pp. 211-220, 1994.

[9] Dustin E., Rashka J., and Paul J., *Automated Software Testing: Introduction, Management, And Performance*, Addison-Wesley Professional, 1999.

[10] Gavetter F., *In: Statistics for Behavioral Sciences,* Wadsworth Publishing, 2010.

[11] Graham D. and Fewster M., *Experiences of Test Automation: Case Studies of Software Test Automation*, Addison-Wesley Professional, 2012.

[12] Grieskamp W., Gurevich Y., Schulte W., and Veanes M., "Generating Finite State Machines from Abstract State Machines," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 4, pp. 112-122, 2002.

[13] Gupta R., Harrold M., and Lou Soffa M., "Program Slicing-Based Regression Testing Techniques," *Journal of Software Testing Verification and Reliability*, vol. 6, no. 2, pp. 83-111, 1996.

[14] Haraty R., Mansour N., and Daou B., *In Volume 3 of Advanced Topics in Database Research*, Idea Group Publishing, 2004.

[15] Harris P. and Raju N., "A Greedy Approach for Coverage-Based Test Suite Reduction.," *The International Arab Journal of Information Technology*, vol. 12, no. 1, pp. 17-23, 2015.

[16] Harrold M. and Lou Soffa M., "Interprocedual Data Flow Testing," *ACM SIGSOFT Software Engineering Notes*, vol. 14, no. 8, pp. 158-167, 1989.

[17] Harrold M. and Rothermel G., "Performing Data Flow Testing on Classes," *ACM SIGSOFT Software Engineering Notes*, vol. 19, no. 5, pp. 154-163,1994.

[18] Leung H. and White L., "A Study of Integration Testing and Software Regression at The Integration Level," *in Proceedings Conference on Software Maintenance*, San Diego, pp. 290-301, 1990.

[19] Lin F., Ruth M., and Tu S., "Applying Safe Regression Test Selection Techniques to Java Web Services," *in Proceedings International Conference on Next Generation Web Services Practices*, Seoul, pp. 133-142, 2006.

[20] Mann M. and Sangwan O., "Generating and Prioritizing Optimal Paths Using Ant Colony Optimization," *Computational Ecology and Software*, vol. 5, no. 1, pp. 1-15, 2015.

[21] Mann M. and Sangwan O., "Test Case Prioritization Using Cuscutta Search," *Network Biology*, vol. 4, no. 4, pp. 179-192, 2014.

[22] Mann M. and Sangwan O., "Test Language Processing: A Novel Approach for Automated Software Testing," *Software Engineering:An International Journal*, vol. 3, no. 2, pp. 29-34, 2013.

[23] Nidhra S. and Dondeti J., "Blackbox and Whitebox Testing Techniques-A Literature Review," *International Journal of Embedded Systems and Applications*, vol. 2, no. 2, pp. 29-50, 2012.

[24] Quick Test Professional: QTP helps documentation. Available with QTP 10.00 Version," 2014.

[25] Ramamoorthy C., Ho S., and Chen W., "On the Automated Generation of Program Test Data," *IEEE Transactions on Software Engineering*, vol SE-2, no. 4, pp. 293-300, 1976.

[26] Rothermel G. and Harrold M., "A Safe, Efficient Regression Test Selection Technique," *ACM Transactions on Software Engineering and Methodology*, vol. 6, no. 2, pp. 173-210, 1997.

[27] Ruth M. and Tu S., "A Safe Regression Test Selection Technique for Web Services," *in Proceedings of 2nd International Conference on Internet and Web Applications and Services*, Morne, pp. 47, 2007.

[28] Ruth M. and Tu S., "Towards Automating Regression Test Selection For Web Services," *in Proceedings of the 16th International Conference on World Wide Web*, Banff, pp. 1265-1266, 2007.

[29] Sangwan O., Bhatia P., and Singh Y., "Radial Basis Function Neural Network Based Approach to Test Oracle," *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 5, pp. 1-5, 2011.

[30] Singh Y., *Software Testing*, Cambridge Press, 2012.

[31] Taha A., Thebaut S., and Liu S., "An Approach to Software Fault Localization and Revalidation Based on Incremental Data Flow Analysis," *in Proceedings of the 13th Annual International Computer Software and Applications Conference*, Orlando, pp. 527-534, 1989.

[32] Tarhini A., Fouchal H., and Mansour N., "Regression Testing Web Services-based Applications.," *in Proceedings of the IEEE International Conference on Computer Systems and Applications*, Dubai, pp. 163-170, 2006.

[33] Vokolos F. and Frankl P., "Empirical Evaluation of The Textual Differencing Regression Testing Technique," *in Proceedings of International Conference on Software Maintenance*, Bethesda, pp. 44-53,1998.

[34] Vokolos F. and Frankl P., *in Reliability, Quality and Safety of Software-Intensive Systems*, Springer, 1997.

[35] Web Link, Online Available at- *http://sourceforge.net/projects/vtigercrm/files/vtiger* CRM, Last Visited, 2005.

[36] Web Link, Online, Available at- *http://www.internetjournals.net/journals/tir/2009/January/Paper% 2006.pdf.*, Last Visited, 2014.

[37] White L. and Leung H., "A Firewall Concept for Both Control-Flow and Data-Flow in Regression

Integration Testing," *in Proceedings of Conference on Software Maintenance*, Orlando, pp. 262-271, 1992.

[38] Willmor D. and Embury S., "A Safe Regression Test Selection Technique for Database-Driven Applications," *in Proceedings of 21st IEEE International Conference on Software Maintenance*, Budapest, pp. 421-430, 2005.

[39] Xu G. and Rountev A., "Regression Test Selection for Aspectj Software," *in Proceedings 29th International Conference on Software Engineering*, Minneapolis, pp. 65-74, 2007.

[40] Xu L., Xu B., Chen Z., Jiang J., and Chen H., "Regression Testing for Web Applications Based on Slicing," *in Proceedings of the 27th Annual International Conference on Computer Software and Applications*, Dallas, pp. 652-656, 2003.

[41] Zallar K., "Are you Ready for the Test Automation Game," *Software Testing and Quality Engineering*, vol. 3, pp. 22-27, 2001.

[42] Zhao J., Xie T., and Li N., "Towards Regression Test Selection for Aspectj Programs," *in Proceedings of the 2nd Workshop on Testing Aspect-Oriented Programs*, Portland, pp. 21-26, 2006.

Mukesh Mann is a Research Scholar in Department of CSE, School of ICT, GBU, India. He is a recipient of UGC-JRF Award -2014, CSIR-SRF Award-2014 and UGC-SRF Award-2016. His areas of research are Computational Intelligence and Software Engineering.

Om Prakash Sangwan is working as an Associate Professor in Department of CSE, GJUST Hisar, India. His areas of research are Software Engineering and Soft Computing.

Pradeep Tomar is an Assistant Professor in Department of CSE, School of ICT, GBU, India. His area of research are Computational Intelligence and Component based Software Engineering.