IDENTIFYING A REGRESSION TEST PRIORITIZATION TECHNIQUE AND PROPOSING A TOOL FOR AUTOMATION FOR TRADE ME WEBSITE

Neenu Ignatious and Shahid Ali

Department of Information Technology, AGI Institute, Auckland, New Zealand

ABSTRACT

This research study is focused on identifying a regression test prioritization technique and suggesting a tool for automating the testing activities for the Trade Me website New Zealand. Identifying the importance of regression testing for a frequently growing application this project is proposed that can be used in similar projects in future. Regression testing is the costliest and time taking part of a software under test. Suggested method can be used for identifying cost and time efficient technique.

KEYWORDS

Regression Test Prioritization, Ant Colony Optimization, Selenium WebDriver

1. INTRODUCTION

Regression testing is the repetitive part of a software under test. It confirms that new defects will not be introduced into the modified code. This report aims at identifying a suitable regression test prioritization technique along with an automation test tool selection for regression automation for the Trade Me website. For attaining the goal, a research-based methodology is followed. Analysed various literature reviews and identified a cost and time efficient technique. The technique identified had some gaps based on our requirement. To be efficient a hybrid approach is suggested to meet the requirements for the Trade Me Website. In order to identify a suitable tool compared two most commonly used automation tool in the market and selected the one that more suits the selected application.

The regression test prioritization technique selected for the project is Ant Colony Optimization technique in combination with customer requirement-based test selection. For proposing a tool two mostly used automation tools in New Zealand is been compared and selected Selenium for the test script automation.

Trade Me is the biggest internet trading website in New Zealand. This site is founded by Slam Morgen a New Zealand Entrepreneur in 1999. It is the fifth most visited site in New Zealand as per internet statistics. The core idea of Trade Me is we connect people and businesses and provide them with tools and information that they needed for a transaction. They are the leading online marketplace and classified advertising platform in New Zealand. In a country with population of 4.9 million, as of August 28, 2019 the number of members' active online is 4687026 (Trade Me Statistics). From the year 1999 to 2019 for two decades Trade Me grown step by step to reach the point as we see now. Lot new functionalities and integrations are implemented year by year. In

Natarajan Meghanathan et al. (Eds) : CSTY, AI, MaVaS, SIGI, FUZZY - 2019
pp. 41-52, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.91404

the year 2000 the feature 'Success fees' introduced 2001 Find someone feature implemented likewise each year has its own new implementation which adds a new functionality for the website. The Trade Me infrastructure is intended to give an adaptable and scalable environment. They operate out in two separate data centres. This provides a redundant infrastructure which allows most of the maintenance to be done during the daytime, without impacting the site.

Trade Me is a rapidly growing website which has its new feature implementation frequently. The traffic in the website is enormous as the quality of the website should not be compromised at any cost. Based on analysis the current regression testing of the company is not efficient to get its maximum benefits. This project identifies proper technique for regression prioritization with the aim of not leaking any of the bugs during regression testing. The regression testing has its great importance it uncovers errors due to change in any of the functionality. This research also includes the proposing a suitable tool for automating the regression suite for the company. The aim of the project is to increase the cost and time efficiency for regression testing and proposing an automation tool for regression. Regression Testing confirms that the previously developed and tested software still functioning the same after a change in the software. Since regression testing is an expensive process, this research is to identify a technique which is cost effective. Understanding the importance of an efficient regression testing for the company this project has the objectives to optimize the regression testing by proper selection, prioritization and automation. The project researches on regression test prioritization techniques and suggest a best technique for the company. The project also aims at suggesting a best tool for automating the regression suite.

This research paper is organized as follow: Section 2 focuses on the literature review of various studies focusing on regression test prioritization techniques. Section 3 of this research is focused on project execution. Discussion to results and research findings are provided in section 4. In section 5 comparative analysis for automation tool selection is provided. Section 6 is dedicated towards the future work recommendations. Finally, in section 7 conclusion to the research is provided.

2. LITERATURE REVIEW

In the past different researches had been conducted for Regression test selection and prioritize regression suite as discussed below:

A study was conducted on regression test prioritization techniques in [1]. In this study they use various approaches like, Fault Based, Coverage Based, Requirement based, Modification based, History based, and Genetic based approaches. As discussed in the literature review it has benefits but this approach is not enough to cover all the important test suites.

A different case study by [2] discusses about 'Retest All' technique which is found not cost effective for 'Trade Me' because of its huge tests count where in which all the test cases been re-executed during the regression testing. Another case study on regression prioritization is conducted by [3] discusses about Data flow analysis-based techniques, Graph walk based techniques mentioned in is only acceptable procedural oriented Programs.

Case study conducted by [4] discussed about prioritization based on fault severity. They extend the code coverage and function coverage prioritization techniques and apply at faults severities. Analysis conducted by [5] on Regression Test Case selection problem compared two methods Random Ordering and Cost-Cognizant Prioritization. They identified Cost Cognizant prioritization can improve test suites rate of fault detection.

In a case study by [6] Hybrid Regression test selection is identified as the best technique for regression test prioritization. In this method it combines two level analysis, method and file for more cost-effective regression test selection.

"Test Case Prioritization for Regression Testing of Service-Oriented Business Applications", by [7] have examined the impact of various artefacts on test case prioritization of service-oriented business applications and illustrated the shortcomings of traditional prioritization techniques.

"Prioritizing Test Cases for Regression Testing", by [2] empirically examined the abilities of various techniques to improve the rate of fault identified. It prioritizes based on specified modified version of a program which is named as version-specific test case prioritization.

For proposal for an automation tool for regression compared various tool in market with the aid of literature reviews. Compared tools Selenium, Sikuli and Watir [8], compared selenium with QTP [9]. Taking a decision on when to automate is also explained in this project paper [10]. In the previous researches, a lot of different aspects of regression test prioritization conducted. My study will be focussed on identifying a time efficient technique that suits for Trade Me website and selection of a tool for automating the regression suite.

3. PROJECT EXECUTION

In this section we will discuss about the project execution for this research. The regression testing usage increases due to the growth in product for under testing. This issue necessitates the importance of selecting test cases effectively, which is a challenging task. That is because it affects the coverage, cost and fault detection during regression testing. As mentioned by several studies, 50 % of a total project cost is invested in testing activities. And out of that 80 % is for regression testing [11].

To maintain the coverage, time and cost effectiveness of test suite in regression testing, the tester can select a small set of test cases that have already been executed on the system under testing (SUT) or prioritize the test cases effectively [12].

Regression test case selection is based on the code modifications of System under Test and selects the test cases which are related to the modifications between the previous and current version of builds. Test suite prioritization rearranges the test cases with the intent to discover faults early from SUT. The main aim of prioritization technique is to increase fault detection ability, disregarding the version of SUT or modifications made to the source code of SUT [13].

Running all the tests cases on the time of regression testing is expensive and time-consuming task. Instead of ‘Retest All’ technique we have various technique like test case selection, test case prioritization and Hybrid approaches. In test case selection technique, instead of re- executing all the test cases a subset of the test cases are selected. They can be classified as 1) Reusable and 2) Obsolete test cases.

Reusable test cases can be used in testing and obsolete are not used in succeeding regression cycles. In Prioritization test cases are prioritized based in certain criteria that which test cases should be executed first and which should be executed later in the run. It does not discard any of the test cases thus avoiding the drawback of test case minimization. In Hybrid approach it is the combination of Test cases selection and prioritization to achieve the advantage of both.

3.1. Test Case Prioritization

The main objectives of test case prioritization are given below:

i. To improve faults detection rate
ii. Early detection of faults which are more risky
iii. Improve reliability of the system

3.2. Test Case Prioritization Techniques

In regression prioritization technique test cases which resulted in maximum fault detection are selected. Test cases with minimum execution time and maximum usage are assigned maximum priority.

Various prioritization techniques are discussed below:

a) Random Prioritization: Here the test cases are randomly ordered without following any criteria.
b) Optimal Prioritization: Optimal ordering of test cases for maximum fault detection
c) Total branch coverage: In this method, based on the total number of branches in the code test cases are prioritized.
d) Additional branch coverage: Test cases are prioritized in the order of coverage of branches that are not yet covered.
e) Fault Exposing potential (FEP) prioritization: Prioritizing based on probability of exposing faults.
   The rate of fault detection is good if followed by the following conditions:
   i. Faults having high risk are revealed at earlier stage
   ii. Critical code sections faults are revealed earlier.
   iii. Confidence is provided in the system reliability
f) Customer Requirement based prioritization:

The prioritization done based on requirement document. In this technique the test cases can rank based on customer assigned priority, requirement complexity and volatility.

3.3. Proposed Method for Project

For Trade Me website ‘Ant Colony Optimization’ algorithm is the proposed method for test case prioritization.

3.3.1. Ant Colony Optimization (ACO)

Ants are tiny blind animals still they can find the shortest path to their food source. They use their antennas and pheromone fluid to communicate with each other. ACO is inspired from the behaviour of live ants on a food hunt to find their optimal path to their food source by maintaining previous information gathered by each ant.

While on food hunt ants follow certain paths. Ants who follow the similar path of other ant recognizes the path by the left behind fluid pheromone by each ant. The optimal path is identified by the pheromone evaluation. Ants take random path and they deposit pheromone. On return they will take the path with more residual pheromone.

Pheromone evaporation rate depends on the length of the path. The more pheromone evaporated the long the path.

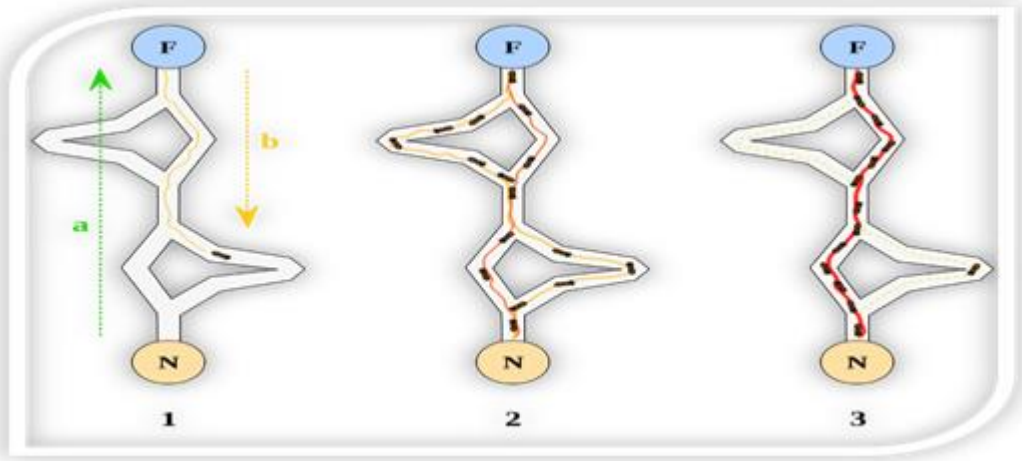

Figure 1: Ants Finding Optimal Path [14]

Figure 1 represents illustrates how ants find the optimal path between their nest and food based on the pheromone deposit. 'F' represents food source and 'N' represent ant nest.

3.3.2. Working of the Proposed System for regression prioritization

The focus of this paper is to optimize the regression testing by prioritizing it with a suitable technique. To achieve this aim, we must consider various matrices as input and sorted after considering several factors. Thus, the efficiency of product will be optimized by reducing the time required to perform the test, reducing the cost and finding maximum faults.

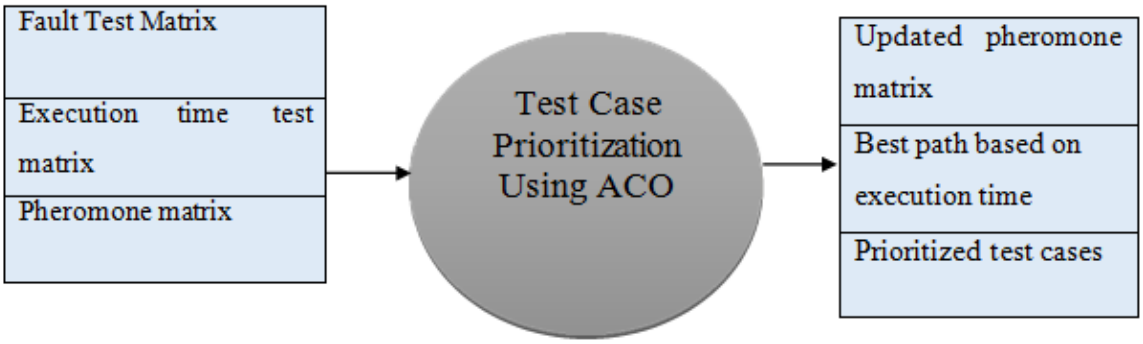

Figure 2: Input and Output matrices of ACO

The input and output matrices that is used in Ant Colony Optimization technique for regression test prioritization is shown in Figure 2. The details about each matrix is discussed below.

Fault Test Matrix represents the number of faults identified by each test case. Execution time test matrix illustrates the time required to execute each test case. Pheromone matrix indicates the pheromone values associated with each test case. Initially it is calculated as 0.

The above mentioned in the paragraph are the inputs for test case prioritization and the output of the system are updated pheromone matrix, best path based on execution time and prioritized test cases. Ant Colony Optimization (ACO) algorithm will work as follows:

i. Initially value for the pheromone is considered as zero.
ii. Iterate for test cases.
iii. First select a test case that cover maximum faults, if it is not covering all faults select next test case that cover remaining faults and terminate the iteration when all faults are found.
iv. Calculate total faults found in each iteration and add in total fault covered matrix.
v. For each path identified calculate the average execution time.
vi. Compare the path based on average execution time and pheromone matrix.
vii. Choose the best path.
viii. Update the pheromone value for the best-chosen path.

3.3.3. Example

Consider a test suite with six test cases in it, covering a total of eight faults. In this section we present the execution of our algorithm in time-based prioritization.

Table 1: Fault Matrix

Test Case	Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Fault 6	Fault 7	Fault 8
TC -001		✓		✓			✓	
TC-002	✓		✓					
TC-003	✓				✓		✓	✓
TC-004		✓		✓				
TC-005			✓			✓		
TC-006	✓						✓	

Faults detected by each test case is represented in Table 1. This information is collected during the previous execution of test cases. The Table 1 illustrates that TC-001 identifies three faults, TC-002 identifies 2 faults, TC-003 identifies the maximum number of faults that is 4, TC-004 identifies 2 faults, TC-005 identifies 2 faults and TC-006 identifies 2 faults.

Table 2: Execution Time Matrix

Test Case	Execution Time
TC-001	6
TC-002	4
TC-003	8
TC-004	9
TC-005	3
TC-006	2

Execution time for each test case is illustrated in the Table 2. TC-001 takes 6-unit time to get executed. TC-002 takes 4-unit of time TC-003 takes 8-unit of time TC-004 need 9 unit of time TC-005 get executed in 3 unit of time and TC-006 takes 2 unit of time to execute. From the table it is shown thatTC-006, TC-005 and TC-002 takes very least time to execute.

From the Table 1 the completed graph Figure 3 is generated which contain all the test cases and paths as shown in the graph below. A single or more ants will start from each vertex of the graph exploring the optimal path. This example illustrates the vertices.

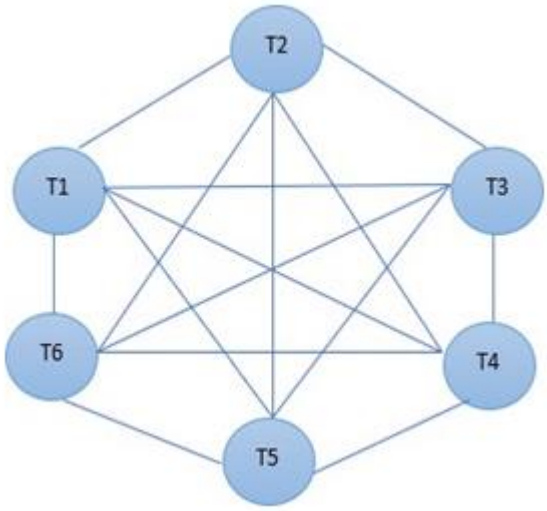

Figure 3: Test Cases and Possible paths

After the first iteration the path of each ant is represented in the above Table 3. It is represented the path followed by each ant the time taken for each ant to identify all faults. After the entire iteration the path selected by ant A4 is optimal because the time for execution was 22 which is minimum when compared to other ants. So, the path T4-T6-T3-T5 is selected and pheromone value is updated. Initially the cost of all the edges is calculated as ‘0’ and for the optimum path ‘1’ is added and 10% of value is lost due to evaporation.

Table 3: Iteration 1

							Total Time
A1	Test Case	T1	T3	T6	T2	T5	
	Time	6	8	2	4	3	23
A2	Test Case	T2	T5	T4	T6	T3	
	Time	4	3	9	2	8	26
A3	Test Case	T3	T1	T5	T6	T4	
	Time	8	6	3	2	9	28
A4	Test Case	T4	T6	T3	T5		
	Time	9	2	8	3		22
A5	Test Case	T5	T6	T1	T3	T4	
	Time	3	2	6	8	9	28
A6	Test Case	T6	T4	T1	T5	T3	
	Time	2	9	6	3	8	27

The updated graph after the first iteration is displayed below in Figure 4. The optimal path selected is T4-T6-T3-T5, in the graph the pheromone value for the path is updated in the figure below.

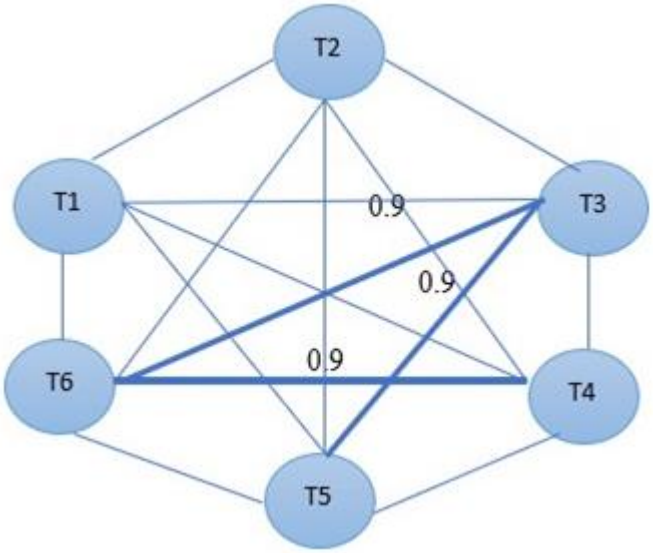

Figure 4: Updated Graph after first Iteration 1

In the next iteration again all the ant searches an optimal path in the graph with available deposit of pheromone. The path selected by ant A3 is optimal out of entire paths. It selects T3-T5-T4

vertex whose execution time is the least 20. The pheromone along that path is updated. When the ant start execution from T3 the highest pheromone at nearby edges are e36=0.9 and e35=0.9. The edges are with the same pheromone so randomly it selects e35=0.9.

Table 4: Iteration 2

							Total Time
A1	Test Case	T1	T3	T6	T4	T5	
	Time	6	8	2	9	3	28
A2	Test Case	T2	T1	T6	T5	T3	
	Time	4	6	2	3	8	22
A3	Test Case	T3	T5	T4			
	Time	8	3	9			20
A4	Test Case	T4	T6	T3	T5		
	Time	9	2	8	3		22
A5	Test Case	T5	T2	T3	T6	T4	
	Time	3	4	8	2	9	25
A6	Test Case	T6	T4	T2	T3	T5	
	Time	2	9	4	8	2	25

The update value of pheromone on this selected path is updated as e35 = (0.9+1)=1.9. The evaporation will be constant 10% so at last pheromone deposit is 1.71. Accordingly, remaining edges are updated. The graph and table are represented in Table 4 and Figure 5.

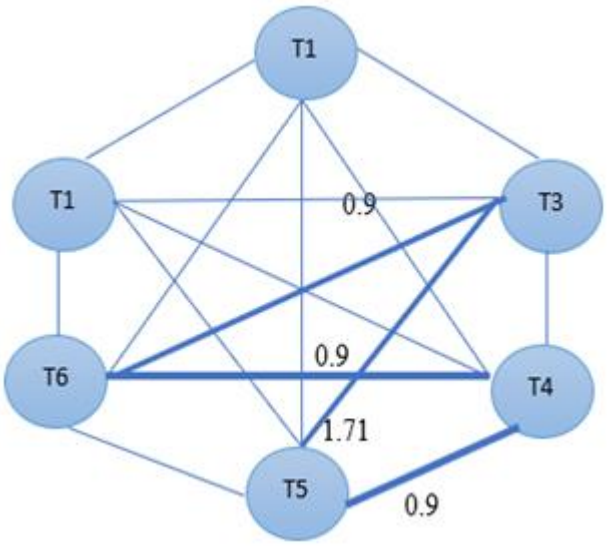

Figure 5: Updated graph after second iteration

According to the selected best path the corresponding pheromone values are updated. Selection of best path is done based on execution time and pheromone value.

For the third iteration it identifies the same optimal path, so the iterations are stopped and finalised the optimum path as (T3-T5-T4). The remaining nodes are taken in any of the order (T1-T2-T6).

4. DISCUSSION

It was found that the result of arising bugs in production is last minute fixes of bugs and the product released after not checking the safety critical features hence selecting test cases for regression testing is an art it should be done with proper care. It requires deep knowledge on the bug fixes and how it affect the system. Effective regression testing can be done by selecting the following test cases:

a. Test cases which identify defects frequently
b. Functionalities which are more visible to the users/customers
c. Test cases that verify the core functionalities are mandatory in the regression suite.
d. Test cases of functionalities which has undergone more and recent changes
e. Test Cases that test the integration.
f. Complex Test Cases
g. Boundary value test cases

Selection of test cases for regression testing depends more on the criticality of bug fixes than the criticality of the defect itself. A minor defect can result in major side effect and a bug fix for an extreme defect can have no or a just a minor side effect. So, the test engineer needs to balance these aspects for selecting the test cases for regression testing.

In order to attain maximum efficiency for a regression test suite prioritization there is no exact answer to it, however, the following key elements need to be considered of:

i. Effective selection of regression suite, the ACO is not applicable for selection of a test case because the aim of regression suite is not for fault detection alone it is to check the overall functionality is not disturbed.
ii. Effective maintenance of regression suite
iii. The selection can be done in hand with customer intervention and an expert from the company who have deep knowledge on project and testing activities of the project.
iv. After this step of identification select the test cases that to be automated for regression.
v. Prioritize the test cases for automation and manual using ACO prioritization techniques.
vi. Finally maintain the regression test suites frequently.

5. COMPARATIVE ANALYSIS FOR AUTOMATION TOOL SELECTION

In order to attain maximum efficiency for regression testing certain test scripts should be automated to identify the repetitive test cases and to automate the test scripts. We have various tools for automating software testing out there and we compared two, Selenium and Micro Focus-UFT and select the best one for automating the prioritized regression suite for Trade Me.

Selenium is selected when compared to the other tool because of the following reasons:

i. Since we must automate the regression testing for Trade Me web application the Selenium tool supports the requirement without any cost.
ii. Selenium is highly extensible and flexible which is selenium's greatest strength.
iii. Can integrate with build tools like Jenkins, Maven.

6. RECOMMENDATIONS

The paper discusses about Ant Colony Optimization for regression test prioritization which gives a result that is almost optimal. The regression test cases should be maintained at regular intervals as the set of test cases would reach a saturation level without finding any faults in the system. Test case selection for regression and prioritization is dependent. If the selection of test cases is inappropriate the result of prioritization will be not efficient.

7. CONCLUSION

In this paper, regression test case prioritization technique is successfully identified. There are various factors based on which test case prioritize can be decided. Test prioritization can strengthen regression testing for finding more severe fault in earlier stages. The goal of this research project was to find the test prioritisation technique which we successfully proposed in section 4.3. Test case prioritization varies from project to project. With earlier prioritization of test cases we can reduce cost, time, effort and maximize customer satisfaction. Using ACO approach one can effectively prioritize regression test suite, with minimum execution time. Hence the proposed algorithm is useful for the Trade Me application and can be used in similar applications. The solution obtained after performing the algorithm is nearest to optimal. ACO is strong & robust as it can lead to better solutions in optimum time. For attaining the maximum out of regression testing after conducting a feasibility analysis certain repetitive tests should be automated. The tool selected for automation is Selenium with TestNG framework for generating proper formatted reports.

REFERENCES

[1] Singh, Y., Kaur, A., Suri, B., & Singhal, S. (2012). Systematic literature review on regression test prioritization techniques. Informatica, 36(4).

[2] Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (2001). Prioritizing test cases for regression testing. IEEE Transactions on software engineering, 27(10), 929-948.

[3] Biswas, S., Mall, R., Satpathy, M., & Sukumaran, S. (2011). Regression test selection techniques: A survey. Informatica, 35(3).

[4] Varun Kumar, S., & Kumar, M. (2010). Test case prioritization using fault severity. IJCST, 1(1).

[5] Ranga, K. K. (2015). Analysis and Design of Test Case Prioritization Technique for Regression Testing. International Journal for Innovative Research in Science and Technology, 2, 248-252.

[6] Zhang, L. (2018, May). Hybrid regression test selection. In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE) (pp. 199-209). IEEE.

[7] Mei, L., Zhang, Z., Chan, W. K., & Tse, T. H. (2009, April). Test case prioritization for regression testing of service-oriented business applications. In Proceedings of the 18th international conference on World wide web (pp. 901-910). ACM.

[8] Singh, I., & Tarika, B. (2014). Comparative analysis of open source automated software testing tools: Selenium, sikuli and watir. International Journal of Information & Computation Technology, 4(15), 1507-1518.

[9] Jagannatha, S., Niranjanamurthy, M., Manushree, S. P., & Chaitra, G. S. (2014). Comparative Study on Automation Testing using Selenium Testing Framework and QTP. vol, 3, 258-267.

[10] Stobie, K. (2009). Too much automation or not enough? When to automate testing. In Pacific Northwest Software Quality Conference.

[11] Chittimalli, P. K., & Harrold, M. J. (2009). Recomputing coverage information to assist regression testing. IEEE Transactions on Software Engineering, 35(4), 452-469.

[12] M. A. Askarunisa, M. L. Shanmugapriya, and D. N. Ramaraj. 2010. Cost and coverage metrics for measuring the effectiveness of test case prioritization techniques. INFOCOMP J. Comput. Sci. 9, 43–52.

[13] S. Yoo and M. Harman. 2007. Pareto efficient multi-objective test case selection. In Proceedings of the 2007 International Symposium on Software Testing and Analysis. ACM, 140–150.

[14] Toksari, M. D. (2016). A hybrid algorithm of Ant Colony Optimization (ACO) and Iterated Local Search (ILS) for estimating electricity domestic consumption: Case of Turkey. International Journal of Electrical Power & Energy Systems, 78, 776-782.

AUTHORS

Neenu Ignatious was born in India. She received my Bachelor of Engineering degree in Information Technology from the Cochin University of Science and Technology, Kerala, in 2012. In the same year I joined as a Quality Assurance Engineer in Infopark, Cochin. Later in the year 2018 I have moved to New Zealand.

Dr. Shahid Ali is a senior lecturer and IT programme leader of information technology at AGI Education Ltd, Auckland, New Zealand. He has published number of research papers in ensemble learning. His expertise and research interests include machine learning, data mining ensemble learning and knowledge discovery.