International Journal of Engineering & Technology, 7 (1.1) (2018) 133-137

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Methodology for regression testing with open source tool

K. Hema Shankari [1*], R. Thirumalai Selvi [2]

[1]*Research Scholar, Bhartath University, Associate Professor, Department of Computer Science, Women's Christian College, India*
[2]*Research Supervisor, Assistant Professor, Department of Computer Science, Govt. Arts College (Men), Nandhanam.*
**Corresponding author E-mail:hems_banu@yahoo.com*

Abstract

The paper describes our methodology for optimizing regression testing that forms a major part of software maintenance. It necessitates the use of an automated testing tool, and we have selected Selenium, an open source tool. For simple projects, a formula is proposed that has been derived through data mining with Selenium. A genetic algorithm is added to this methodology for industry based projects, where the test cases are so large that they have to be grouped as Test Suites; this algorithm reconfigures Test suites in each cycle of regression testing. Commonly used APFD metric ignores fault severity but is included in our formula; this severity is determined by professional testers. The use of ANN to amend severity without manual intervention enhances the genetic algorithm. Tables presented in the paper are from both simple and industry projects. Comparison is made with IBM'S RFT, a proprietary tool for automated testing.

Keywords: *Regression testing, genetic algorithm, selenium tool, APFD metric, ANN.*

1. Introduction

As distinct from unit testing, integration testing, and user acceptance testing, that take place during development phase, regression testing takes place during the maintenance phase. It is estimated that some sixty percent of project cost is associated with maintenance. A great deal of research has therefore been conducted about regression testing both in academia and industry. An important milestone is the availability of software tools that support regression testing. While RFT is an IBM tool that integrates with its Rational Manager, Selenium is an open source tool. Initial automation was confined to capturing key strokes of testers in a computer terminal; there has been rapid progress in building a comprehensive database of test data with these tools and consequent data mining. This is even more critical with agile software development methodology. Our research makes extensive use of such databases. For projects completed by us, we dealt with test cases and detected faults. For severity rating, we relied on our testers. The severity classification was kept unchanged in subsequent cycles of maintenance. The priority of test cases was established with a formula that is explained later. In our discussions with industry, it became apparent that the number of test cases is so large that they have to be grouped as 'Test Suites'. Consequently, the formula proposed by us must be applied to Test Suites rather than to test cases. Another important observation was the industry's focus on building business models to describe software functionality, and generation of Test Suites based on business model. This means that a manufacturing industry will approach regression testing differently to retail industry. Software development companies in India like Infosys and TCS divided their organizations into so called 'verticals'. This made our task difficult since we attempt to develop a generic approach to regression testing irrespective of the industry. The next sections of this paper elaborate the building of business model and the methodology.

2. Metrics

In relapse testing, reusing of utilized cases can significantly enhance test effectiveness, and decrease time and duplication of exertion. Along these lines, there is an enormous experiment library at the steady stage. It lists all the utilized cases in inventory to connect the particular cases with related organizations, and offices the reference of cost-evaluation display and the programmed age of the test script.
Though benefits of the business frameworks progress for the change for demands, What's more with the transforms for framework upkeep and different reasons; whether new forms of the programming need aid generated Toward those improvement department, execution steps relapse trying from claiming are as takes after:

1. Examine What's more dissect those sourball codes in the new version, Furthermore behavior dissection for transforms bases on the requisition model, programmed recognizing framework changes;.
2. Examination for transform effects Investigation faultlessly pointed crazy the scopes for practical benefits of the business straightforwardly alternately by implication impacted Eventually Tom's perusing An change about rendition.
3. For those provision for benefits of the business rules, the relapse test ranges would controlled Eventually Tom's perusing masters Furthermore investigators.
4. Test suited will be created in the appraisal model for expense and risk, Furthermore it will a chance to be compacted with streamlining calculation.
5. Finish programmed trying Eventually Tom's perusing denying utilized test instances in the library alternately Creating new instances.

Those APFD metric simply exhibited depends around two assumptions: (1) at faults have equivalent severity, Also (2) the sum test cases bring equivalent expenses. Over practice, however, there need aid situations clinched alongside which these presumptions don't hold: instances On which faults fluctuate done seriousness and test situations change over expense. Clinched alongside such cases, those APFD metric might furnish unsuitable outcomes.

1. Average Percentage Block Coverage (ABC).
   This measures those rate In which a prioritized test suited blankets the obstructs.
2. Average Percentage Decision Coverage (ADC).
   This measures the rate at which a prioritized test suited blankets those choices (branches).
3. Average Percentage Statement Coverage (ASC).
   This measures the rate at which a prioritized test suite covers the statements.
4. Average Percentage Loop Coverage (ALC).
   This measures those rate during which a prioritized test suited blankets those proclamations.
5. Average Percentage Condition Coverage (ACC).
   This measures the rate toward which a prioritized test suited blankets the loops.
6. Problem Tracking Reports (PTR) Metric
   The PTR metric is An alternate approach that those viability of a test prioritization might a chance to be investigated.

Review that an viable prioritization system might put test cases that are well on the way on recognize faults toward the start of the test succession. It might make useful to ascertain the rate of test cases that must a chance to be run in front of at faults bring been uncovered. PTR may be ascertained as takes after:

$$Ptr(t,p) = nd/n$$

Give t - be those test suited under evaluation, n - those aggregate number for test instances in the aggregate amount about test situations required with identify at faults in the system under test p.

3. Testing tools

A. RFT tool

Normal utilitarian analyzer product is an robotized device which gives testers with robotized trying competencies for utilitarian testing, relapse testing, GUI trying Furthermore information determined trying.

As a robotized trying tool, RFT need a few Characteristics below:

1. Provide robust testing support for Java, Web 2.0, SAP, Siebel, terminal-based and Microsoft Visual Studio .NET Windows Forms applications
2. Perform story board testing to combine natural language test narrative with visual editing through application screen shots.
3. Use keywords to bridge the gap between manual and automated testing
4. Manage validation of dynamic data with multiple verification points and support for regular expression Pattern matching
5. Reduce rework, minimize the rerecording of scripts, and reduce script maintenance

B. Selenium tool

For web applications, we have portable software called Selenium. We use this tool for both recording and subsequent playback; for authoring test cases we do not need to learn Selenium IDE; we need, however, to learn a test-specific language Selenese; with this, we can write tests in a number of popular programming languages, including C#, Groovy, Java, Perl, PHP, Python, Ruby and Scalar. The tests can then be run against most modern web browsers. Selenium deploys on Windows, Linux, and OSX platforms. It is open-source software, released under the Apache2.0 license, and can be downloaded and used without charge.

Selenium is at present the most powerful freeware of open source automation tool. It is developed by Jason Huggins and his team. This is release under the Apache2.0 license and can be downloaded and used without any charge.

Selenium is easy to get started with for simple functional testing of web application.

It supports record and playback for testing web based application.

Selenium supports multithreading feature i.e. multiple instance of script can be run on different browsers.

Test Maker integrates Selenium to provide the important features and benefits:

1. Selenium supports languages such as Java, Perl, and Python, C #Ruby, Groovy, Java Script, and VBScript etc.
2. Selenium support many operating systems like Windows, Macintosh, Linux, Unix etc.
3. Selenium supports many browsers like Internet explorer, Chrome, Firefox, Opera, and Safari etc.
4. Selenium can be integrated with ANT or Maven kind of framework for source code compilation.
5. Selenium can be integrated with Test NG testing framework for testing our applications and generating reports.
6. Selenium can be integrated with Jenkins or Hudson for continuous integration.
7. Selenium can be integrated with other open Source tools for supporting other features
8. Selenium can be used for Android, IPhone, Blackberry etc. based application testing.
9. Selenium supports very less CPU and RAM consumption for script execution.
10. Selenium comes with different component to provide support to its parent which is Selenium IDE, Selenium Grid and Selenium Remote Control (RC).

4. Simple case study

This was developed in Java by students and tested using Selenium Tool Tester. Six test cases were used to test its functionality and they were prioritized by using the formula for test case ranking:

$$TCR = (S * N) / time \text{ --- } (2)$$

In this formula, N is the number of faults detected while using the test case, time is the number of minutes of testing with this test case, and S is the severity value of the fault detected (as assigned by the tester). Where more than one fault is detected, a weighted summation is used in the formula.

Full explanation for the formula is given in our previous paper [1] presented at the Multi Conference of Engineers and Computer Scientists 2016. There were 6 test cases and 8 faults were detected during these tests.

The table gives in binary format which of the faults were detected during the six tests (zero representing absence of detection and one representing detection).

However, once risk severity and time for testing are
included, the priority sequence became T4, T2, T5, T1, T6, T3 as explained in our paper [1].

Factors consider for new proposed approach

Three factors that were considered for prioritization [1] include Rate of Fault Detection, Percentage of Fault Detected, and Risk Detection Ability [7]. To every fault a Risk value has been allocated based on a 10-point scale expressed as

Very High Risk: RV of 10

High Risk: RV of 8
Medium Risk: RV of 6
Less Risk: RV of 4
Least Risk: RV of 2.

For test case Tk, RDAk have been computed using severity value Sk ,Nk is the number of defects found by Tk, and time k is the time needed by Tk to find those defects. The equation for RDA can be expressed as:

$$RDAk = (Sk * Nk) / time\ k \quad --- (3)$$

A. Test Case Ranking

For ranking the test cases, all we need to do is sum up the three different components that are RFD, PFD and RDA. This is given below in the form of an equation:

$$TCRk = RFDk + PFDk + RDAk \quad --- (4)$$

Table I: Fault Matrix

Faults / Test cases	F1	F2	F3	F4	F5	F6	F7	F8
T1	X	X		X	X	X	X	X
T2	X							
T3	X				X			
T4		X	X				X	
T5				X		X		X
T6		X		X		X		

In Table I, the regression test suite T contains six test cases with the initial ordering as T1, T2, T3,T4, T5, and T6

Table II: Binary Representation of Test Cases

Test cases	Binary form
T1	11011111
T2	10000000
T3	10001000
T4	01100001
T5	00010101
T6	01010100

Table III: Number of Faults, Execution Time and Risk severity of Faults for Every Test Case

Test cases	No of faults covered	Execution time	Risk severity
T1	2	12	8
T2	3	14	10
T3	1	11	4
T4	4	10	20
T5	2	10	12
T6	2	13	6

This Table III assumes a priori knowledge of the faults detected by *T* in the program *P*.

Table IV: RFD, PFD, RDA for Test Cases T1..T6

Test cases	RFD	PFD	RDA
T1	1.66	2	1.333
T2	2.142	3	2.142
T3	0.9	1	0.3636
T4	4.0	4	8
T5	2.0	2	2.4
T6	1.538	2	0.923

The values of RFD, PFD, and RDA for testcasesT1..T6 are calculated by using (1), (2)and (4), respectively. Table IV represents the values for all three factors which are RFD, PFD, RDA for tes case T1.. T6 respectively.

Table V: Test Case Ranking for T1..T6 Respectively

Test cases	Test case ranking TCR=RFD+PFD+RDA
T1	4.993
T2	7.284
T3	2.263
T4	16
T5	6.4
T6	4.461

Table VI: Test Cases Ordering for Proposed Approach and Previous Work

Test cases	Prioritized order
T1	T4
T2	T2
T3	T5
T4	T1
T5	T6
T6	T3

5. Industry based case study

APGPCL the First Gas Power Plant in A.P. and South India APG PCL is the first gas based power plant to be set up in Andhra Pradesh and South India– attribute to the pioneering efforts of APSEB and the entrepreneurial spirit of Industries in Andhra Pradesh. APGPCL is an innovative business model of Public-Private Partnership. APGPCL is the lowest cost Gas based electricity generating station in the country. Both Stage-I and Stage-II Plants of APGPCL were built ahead of the scheduled time and within the estimated costs.

This research endeavor displays an intricate industry requisition. They exemplify, dependent upon An cement the event study, how test particular architects camwood Right away fill in with the coordinated test nature's domain.

Here, the test cases were made into several sets, each set of test cases being called a Test Suite. So, while Tn is Test Suite n, tjk is the test case j in Test Suite k. For prioritization, a genetic algorithm was used.

The process includes

Step 1: Organize manually the test cases as sets in Test Suites
Step 2: Identify the scope of the next release and determine which change request will be included in the next build.
Step 3: Document the system requirements, functional requirents and implementation plans
Step 4: Implement the change
Step 5: Test or verify the change
Step 6: Release

Improvised Industry Oriented Genetic Algorithm f or Regression Test Case Prioritization

Input: Test suite TK and test case ranking (TCR) for every test case are inputs of the algorithm.
Output: Prioritized order of test cases
Algorithm: A web based project had a total of 244 test cases. Here, these test cases were made into several sets, and each set of test cases is called a 'Test Suite'. So, while Tn is Test Suite n, tjk is the test case j in Test Suite k. For prioritization, a genetic algorithm was used.

1. Organize, manually, test cases as sets in Test Suites
2. Carry out Regression Testing, tracking defects, measuring test time, and assigning severity manually (very high risk = 10 etc. to least risk = 2)
3. Select Test Suites for mutation based on the formula
4. Perform mutation of selected Test Suites
5. Repeat steps 2, 3,4 Only the top 80% of Test Suites were selected for mutation, bottom 20% being left untouched. Mutation involved a simple (and random) swap of test cases between pairs of Test Suites. So, the genetic algorithm did not increase the number of Test Suites or the number of test cases, but merely the way the grouping was done. Another approach is mentioned in [9].

There are 244 cases for this Industry case study in which each test case assigned a priority in which Priority number 5 has the least priority and Number 1 has the Highest priority and there are 34 functionality.

Table VII: 34 Functionality with no. of Test Cases and Priority

s.no	Functionality	No. of Test case	Test case priority
1	Login Access password	2	5
2	Reset Accessing	4	3
3	Control Panel	1	2
4	Menu section	22	4
5	Home page control	1	3
6	General Menu - Page Content	9	3
7	About Us	7	3
8	Contact Us	7	3
9	Opportunities	7	3
10	Add News & Events	17	3
11	Board of Directors	9	3
12	Upload Photos to folder	9	3
13	Gallery	8	2
14	Photo Folder	6	3
15	Directors Login	8	3
16	Directors Info	18	3
17	Directors Involved	17	3
18	Directors Meeting Info	10	3
19	Committee	3	4
20	Settings	1	2
21	Change Profile	6	4
22	Change Password	8	3
23	Partners in Power	12	3
24	Partners	9	3
25	Partners Login	10	3
26	Balance Sheet	4	3
27	General	1	3
28	MOU	3	3
29	Surplributionus Power Disturbtion	2	3
30	Daily Generation	10	3
31	Monthly Generation	6	4
32	Monthly Actual Generation	1	4
33	Expected Generation	6	3
34	Actual Power Generation	7	3

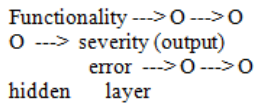
6. Use of neural network in regression testing

Neural Network is a computational intelligence technique inspired by biological nervous system. They are information processing. A paradigm and are use for pattern recognition. Neural networks are physical cellular systems which can acquire, store and process the experiential knowledge .Like humans, neural networks learn by examples and their past experiences. They are expert in deriving meaning form imprecise data and extracting patterns. They are adaptive to the surroundings. They do not use any algorithmic approach to solve a problem to train the neural network model is as follows

Step 1: Data should be collected
Step 2 : A Network should be created
Step 3: Configure the network
Step 4: Initialize the weights and biases
Step 5: Train the network
Step 6: Validate the network
Step 7: Use the network.

A. Defect severity automation

Even with automated testing tools like Selenium, some amount of manual intervention is required. Our attempt, using genetic algorithm, is an effort to minimize such intervention in subsequent cycles of regression testing. For this reason, the defect severity was kept unchanged. This section of the paper describes our attempt to modify defect severity in each cycle of regression testing with the help of an Artificial Neural Network (ANN). The initial ANN for our case study has just 2 inputs --- functionality code and error code. It has just one hidden layer.



For the project completed there were 2 functionality codes as shown in the table below.

Code | Functionality

1 | Change password

2 | Monthly generation

The error code was assigned zero when the actual output did not match predicted output; otherwise, the run time error code of the Java Virtual Machine was assigned.

The table below presents the results of ANN as compared with manually assigned severity. (For each of the 8 faults, the remaining 7 faults were used to train the ANN.)" Each Output has defined category.

Table VIII

ANN OUTPUT	Tested application output	
	CORRECT	WRONG
Correct	1 True Positive	2 True Negative
Wrong	4 False Negative	3 False Positive

Table VIII displays the four possible categories where each output can be placed. Since the ANN is only an approximation of the actual system, some of its outputs may be incorrect. On the other hand, the tested application itself may produce errors, which is the main reason for the testing process. If the ANN output is correct while the output of the tested application is wrong, the evaluation of the comparison tool is classified as being a true negative or a category of 2.

7. Conclusion

A simple case study in regression testing is first discussed where the formula derived through data mining is sufficient to prioritize test cases. For a project completed by our students, the methodology is illustrated through tables. However, for industry based projects, the test cases are so large that they need to be grouped as Test Suites. A genetic algorithm described in this paper is used to re-configure Test Suites, comprising of test cases, to enhance prioritization. Since defect severity is needed for this prioritization, an attempt is made to automate this step through ANN with just two inputs (functionality code and error code). The methodology proposed by us requires use of automated testing tool, and the use of Selenium open source tool is demonstrated.

References

[1] HemaShankari K, ThirumalaiSelvi R & Balasubramanian NV, "Industry Based Regression Testing Using IIGRTCP Algorithm and RFT Tool", *Lecture Notes in Engineering and Computer Science: Proceedings of The International Multi Conference of Engineers and Computer Scientists*, pp.473-478, (2016).

[2] Rothermel, G, Untch, R, Chu, C & Harrold, M, "Test case prioritization: An empirical study", *IEEE International conference, on Software Maintenance*, pp.179-188, (1999).

[3] Pravin A & Srinivasan S, "An Efficient Algorithm for Reducing the Test Cases which is Used for Performing Regression Testing", *2nd International Conference on Computational Techniques and Artificial Intelligence*, pp. 194-197, (2013).

[4] Elbaum S, Malishevsky A & Rothermel G, "Prioritizing test casesfor regression testing", *Proc. ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp.102–112, (2000).

[5] Wong W, Horgan J, London S & Agrawal H, "A study of effective regression testing in practice", *Proc. of the EighthIntl. Symp. On Softw*, pp.230-238, (1997).

[6] Beena R & Sarala S, "Code Coverage Based Test Case Selection And Prioritization", *International Journal of Software Engineering & Applications*, Vol.4, No.6, (2013).

[7] Kavitha R & Sureshkumar N, "Test Case Prioritization for Regression Testing based on Severity of Fault", *International Journal on Computer Science and Engineering (IJCSE)*, (2010).

[8] Musa S, Sultan A, Md AGA & Baharom S, "A regression test case selection and prioritization for object-oriented programs using dependency graph and genetic algorithm", *Research Inventy: International Journal of Engineering and Science*, Vol.4, No.7, pp.54-64, (2014).

[9] Sujatha M.K & Varun, K, "Requirements based Test Case Prioritization using Genetic Algorithm", *International Journal of Computer Science and Technology*, Vol.1, No.2, pp.189-191, (2010).

[10] Farooq QUA, Iqbal MZZ, Malik ZI & Nadeem A, "An approach for selective state machine based regression testing", *Proceedings of the 3rd International Workshop on Advances in Model-based Testing*, pp.44–52, (2007).

[11] Rothermel G, Untch R, Chu C & Harrold MJ, "Test case prioritization: an empirical study", *Testing European Journal of Scientific Research*, Vol.55, No.2, pp.261-274, (2011).

[12] Elbaum S, Malishevsky A & Rothermel G, "Incorporating varying test costs and fault severities into test case prioritization", *IEEE Proceedings of the 23rd International Conference on Software Engineering*, pp. 329-338, (2001).