Journal of Computer Engineering
ISSN 10003428

Original Research Article
Vol 9, Issue 10, 2020 | ISSN: 1000-3428

Evaluating Regression Testing Tools with Genetic Algorithm Optimization

Dr. Meera Nalini, Dr. Rukmini Srinivasan, Dr. Natarajan Kumar

Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India; Department of Information Technology, Anna University, Chennai, India; Department of Artificial Intelligence, Vellore Institute of Technology, Vellore, India

***Abstract*—**The present paper addresses to the research in the area of regression testing with emphasis on automated tools as well as prioritization of test cases. The uniqueness of regression testing and its cyclic nature is pointed out. The difference in approach between industry, with business model as basis, and academia, with focus on data mining, is highlighted. Test Metrics are discussed as a prelude to our formula for prioritization; a case study is further discussed to illustrate this methodology. An industrial case study is also described in the paper, where the number of test cases is so large that they have to be grouped as Test Suites. In such situations, a genetic algorithm proposed by us can be used to reconfigure these Test Suites in each cycle of regression testing. The comparison is made between a proprietary tool and an open source tool using the above-mentioned metrics. Our approach is clarified through several tables.

I. Introduction

A S 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 60% 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 presented 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 developed by us for regression testing. A case study is discussed to illustrate our approach. The genetic algorithm described in this paper is meant for larger industrial projects, and is thus a proposal only.

II. Issues for Regression Testing in Industry Application

A. Issues

Large scale business systems cannot accurately define changes made to it. The number of test cases expands dramatically with combination of several parameters [2]. Complete regression testing is thus impractical [3], [4]. Automated tools are helpful. But, they do not come with any Regression Test Framework. Testing Tools integrate with Software Development Frameworks like IBM Eclipse. Functional testing, based on specific business needs of industry, does require considerable human intervention. With limited time and resources, a methodology is needed for test planning based on risk assessment and cost estimation so that revisions can be completed by given deadlines [8].

1. Methodology

Test models that we surveyed in the literature make too much reliance on software development process. They do not consider the uniqueness of regression testing. Different from unit testing, integration testing, and performance testing, regression testing is based primarily on accumulation of data and careful analysis subsequently. Regression testing is also cyclic.

To set up an application description model, expert knowledge of the industry is needed [5]. Rule based engine is prevalent in the industry. Risk assessment model is also concurrently needed. These drive regression test implementations. The steps in Regression Testing include

How to cite: Dr. Meera Nalini, Dr. Rukmini Srinivasan, Dr. Natarajan Kumar (2020). Evaluating Regression Testing Tools with Genetic Algorithm Optimization. Journal of Computer Engineering, 9(10), 31–38. DOI: https://doi.org/10.5281/zenodo.19342176

scanning and analysis of source codes in the new version, analysis of changes based on business model, impact analysis, determination of test ranges by expert analysts, grouping of test cases as Test Suites, risk analysis, cost estimation, and supplementing existing test cases in the library with new ones.

2. Limitations of the APFD Metric

As pointed out in our previous paper [1], the APFD metric relies on two assumptions; namely, all faults are treated equally in terms of severity, and the costs associated with test cases are the same. Neither of these assumptions are valid leading to unsatisfactory results. The various alternative metrics have also been included in [1] such as Average Percentage Block Coverage, Average Percentage Decision Coverage, Average Percentage Statement Coverage, Average
Percentage Loop Coverage, and Average Percentage Condition Coverage. The Problem Tracking Reports (PTR) Metric is another way that the effectiveness of a test prioritization may be analyzed. PTR is calculated as:

$$Ptr(t,p) = nd / n \quad (1)$$

Let t- be the test suite under evaluation, n- the total number of test cases in the total number of test cases needed to detect all faults in the program under test p.

III. Regression Testing Methods For Industry-Oriented Application

For regression testing of industrial applications, we need a platform that supports decision making; the platform should include, but not limited to, business system rules, description of the application (Use Case diagrams), analysis of changes based on requirements, cost assessment, risk assessment, and management of test cases.

A. Extraction and Loading of Business Rules

Business rules are defined as constraints and norms or business structure and operation. They are important resources for enterprise business operations and management decisions. For more details on sources for business rules, see [1]. All these rules are added to a rule based engine that basically operates on If-Then-Op format. In addition, there are software tools that examine source code and highlight changes made in the new version.

By and large, software development in these days takes place under the umbrella of a Development Environment (Haskell for example uses Cabal). Unit testing is included in such an environment (HUnit is Haskell testing platform). Regression testing too makes use of IDE (Haskell uses QuickCheck for such testing, and it includes randomized testing for enhancing tester's productivity). The Eclipse Environment from IBM is comprehensive in that it spans several programming languages. We thus see the dichotomy between business model on one hand and programming support environment on the other hand, making it difficult for researchers to strike a common ground [6].

IV. Case Study

A. 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:

$$TCR = (S * N) / time \quad (2)$$

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). When 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 six test cases and eight 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].

B. 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 and 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:

$$RDA_k = (S_k * N_k) / time\ k$$

(3) *B. 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:

$TCR_k = RFD_k + PFD_k + RDA_k$ (4) *C. IIGRTCP (Improvised Industry Oriented Genetic Algorithm for Regression Test Case Prioritization)*

The proposed prioritization technique can be expressed as

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].

V. Selenium Tool

A. Features of Selenium Tool

For web applications, we have a 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 Scala. 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, 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).

VI. Experiment and Analysis

It is best to describe the analysis in the form of tables.

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, 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 example in 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	2	1.333
T2	1.285	3	2.142
T3	0.54	1	0.3636
T4	2.4	4	8
T5	1.2	2	2.4
T6	0.9	2	0.923

The values of RFD, PFD, and RDA for test cases T1 ... 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 testcaseT1 .. T6, respectively.

TABLE V
TEST CASE RANKING FOR T1 .. T6 RESPECTIVELY

Test cases	Test case ranking TCR=RFD+PFD+RDA
T1	4.33
T2	6.427
T3	1.909
T4	14.4
T5	5.6
T6	3.8

In Table V, the test case ranking for each test case is calculated.

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

In Table VI, the Test cases are arranged in decreasing order of TCR for the purpose of execution. Test cases are ordered in such a manner that those with greater TCR value execute first.

A. Industry Based Case Study

APGPCL, the First Gas Power Plant in A.P. and South India APGPCL is the first gas based power plant to be set up in Andhra Pradesh and South India – a tribute 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 case study presents a complex industry application. They exemplify, based on a concrete case study, how test engineers can now work with the Integrated Test Environment.

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 requests will be included in the next build.

Step 3. Document the system requirements, functional requirements, functional specification, and implementation plans for each grouping of change requests.

Step 4. Implement the change.

Step 5. Test or verify the change. Unit testing is done by the person whom and the change, usually the programmer. Function testing tests a functional area of the system to see that everything works as expected.

Step 6. Release.

Only the top 80% of Test Suites were selected for mutation, the remaining 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.

The table give the number of faults detected, execution times, and weighted risk severity for the Test Suites when Regression Testing was done the first time, and Test Case Ranking of the Suites based on the formula presented in the industry case study.

For want of space, only six of the 244 test cases, randomly selected, are presented in the Table VII.

TABLE VII
TEST CASES FOR THE INDUSTRY CASE STUDY

Test Case	Test Case Name	Description	Test Case Priority
1	Login Access – APGPCL Control Panel – Valid Credentials	Control Panel of APGPCL application with valid credentials	5
2	Login Access – APGPCL Control Panel – Invalid Credentials	Control Panel of APGPCL application with invalid credentials	5
3	Accessing APGPCL Webpage Control Panel Application	Accessing the Actual APGPCL application from the Control Panel Application	2
4	Password Reset Page -APGPCL Control Panel	Password reset for the APGPCL Control Panel	3
5	Password Reset Page-APGPCL Control Panel – With Valid email	Password reset for a valid APGPCL User	4
6	Password Reset – APGPCL Control Panel - With Invalid email	Password reset for an invalid APGPCL User	3

B. Comparison with the Previous Work

TABLE VIII
APFD % FOR RFT TOOL AND SELENIUM TOOL

Prioritization Technique	APFD %
IIGRTCP with RFT Tool	88%
IIGRTCP with Selenium Tool	91%

In this section, the proposed prioritized order is compared with the previous work.

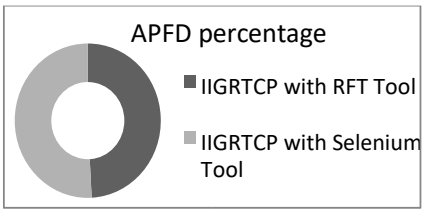

Fig. 1 APFD percentage for RFT Tool and Selenium Tool

VII. CONCLUSION

The difference between Industry approach and that adopted by researchers has been highlighted; while business model forms the backbone for regression testing, data mining of test data gathered with software tools is used to test hypotheses of researchers. For simple case studies, use of a formula proposed by us is sufficient to prioritize test cases; for industry-sized software, a genetic algorithm developed by us is needed in addition to the reconfiguration of Test Suites comprising of test cases before our formula can be applied. We need to use software tools to make use of our proposal; we have discussed two of them - IBM's RFT and open source Selenium tool. For quantifying data, metrics are needed, and this aspect has been elaborated.

There is an urgent need to introduce automation for assigning weight to severity of defects discovered during testing. Work is in progress to use artificial neural networks, but this must be done sector by sector. The finance sector seems to be the best starting point since software for this sector is constantly undergoing modification not only to meet increasing demand from bank clients but also the everchanging government legislation. Training of ANN may be done by different professional testers, and it would be interesting to see if the weighting assigned by the software for defect severity varies with training.

Other approaches to grouping test cases under Test Suites are also under scope for future research [10]-[12].

REFERENCES

[1] K Hema Shankari, R. Thirumalai Selvi, and N. V. Balasubramanian, "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 2016, 16-18 March, 2016, Hong Kong, pp473-478.

[2] G. Rothermel, R. Untch, C. Chu and M. Harrold, "Test case prioritization: An empirical study," In Software Maintenance, 1999. (ICSM' 99) proceedings. IEEE International conference, on pages 179188 IEEE, 1999.

[3] A. Pravin and Dr. S. Srinivasan, "An Efficient Algorithm for Reducing the Test Cases which is Used for Performing Regression Testing," 2nd International Conference on Computational Techniques and Artificial Intelligence (ICCTAI'2013) March 17-18, 2013.

[4] S. Elbaum, A. Malishevsky, and G. Rothermel, "Prioritizing test cases for regression testing," Proc. The 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis, Portland, Oregon, U.S.A., August 2000, 102–112.

[5] W. Wong, J. Horgan, S. London and H. Agrawal, "A study of effective regression testing in practice," In Proc. of the Eighth Intl. Symp. On Softw Rel. Engr., pages 230–238, Nov. 1997.

[6] R. Beena, Dr. S. Sarala, "Code Coverage Based Test Case Selection And Prioritization," International Journal of Software

Engineering & Applications (IJSEA), Vol. 4, No.6, November 2013.

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

[8] Samaila Musa, Abu Bakar Md Sultan, Abdul Azim Bin Abd Ghani, Salmi Baharom, "A Regression Test Case Selection and Prioritization for Object-Oriented Programs using Dependency Graph and Genetic Algorithm" Research Inventry: International Journal of Engineering And Science Vol.4, Issue 7 (July 2014), PP 54-64 Issn (e):2278-4721, Issn (p):2319-6483.

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

[10] Q.-u.-a. Farooq, M. Z. Z. Iqbal, Z. I. Malik, and A. Nadeem. An approach for selective state machine based regression testing. In Proceedings of the 3rd International Workshop on Advances in Modelbased Testing (A-MOST 2007), pages. 44–52, New York, NY, USA, 2007.ACM.

[11] G. Rothermel, R. Untch, C. Chu, and M. J. Harrold. "Test case prioritization: an empirical study" Testing European Journal of Scientific Research, ISSN 1450-216X Vol.55 No.2 (2011), pp.261-274.

[12] S. Elbaum, A. G. Malishevsky and G. Rothermel, (2001), "Incorporating varying test costs and fault severities into test case Prioritization", 23rd International Conference.