Proc. of The Fifth Intl. Conf. On Advances in Computing, Electronics and Electrical Technology - CEET 2016
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-087-3 doi: 10.15224/ 978-1-63248-087-3-65

Using Textual Similarity for Test Suite Prioritization

Iyad Alazzam

Abstract—**Techniques in test suite reduction concentrated on different testing activities in general and regression testing in specific. Regression testing is typically performed after updating or maintaining the software in order to verify that the occurred changes do not incur any problem. Executing all test suites after any modification increases the cost of testing. Thus reduction in test suites resolves this problem by including only relevant test cases. In this paper, a new technique is proposed and presented to reduce the test suite by calculating the similarity between test cases and the source implemented through using Latent Semantic Indexing (LSI) .Moreover the proposed technique is evaluated through calculating the code coverage using EclEmma.**

Keywords—**Regression testing, test case, Latent Semantic Indexing (LSI), and test suite prioritization**

I. Introduction

Releasing a new oran updated version of software increases the percentage of estimated revenue and profits [1]. Nonetheless, in advance and before releasing the software, the new release must be tested in order to ensure that it meets its specifications that have been required and proposed by stakeholders. The testing process is an expensive one which may exceed more than fifty percent of the total cost of the software development process [2]. Moreover exhaustive testing is impossible especially when the software complexity is high. Testing cost grows exponentially along with the product size [3]. Hence, the test plan should include how to prioritize test cases' selection specially when testing resources are limited. Re-execution of all test suites after modifying software takes a significant time. For one software that consists of twenty thousand lines of codeit requires more than one month to execute the whole test suite[4][5]. Therefore, selecting a subset of test suites which is effective and efficient in detecting defects is required. Test prioritization is one of the main activities in regression testing in addition to test cases' selection and reduction [6]. Test prioritization process prioritizes the test suites to be executed which helps the testing team to determine which test suite should be executed first. The test suite with high priority tends to be executed first. Many prioritization techniques are based on previous awareness of the defects and which test cases have the ability to detect these faults. This leads to optimal prioritization which is considered a perfect method in optimization from theory perspective [6]. However, it needs to take into account all likely ordering and arranging of test cases where the run time is going to be high in the worst case depending on the size of the test suite. This paper presents a new technique in prioritizing test suites. The technique is based on finding the strength of the test suite according to the source code of the System Under Test (SUT). The strength (i.e. the quality of the test suite selection) is measured through finding the similarity among test cases in the test suite with the methods in source code. Latent Semantic Indexing (LSI) is used to measure the similarity between test cases and source code. The rest of the paper is organized as the following: next section provides the current research in the area of test suite prioritization. Section three explains and illustrates the textual similarity based test suite reduction approach, and section 4 shows the experiments and the analysis of the conducted experiments. Section 5 concludes the paper.

II. Related Work

Many researches are presented about test cases reduction assessment and prioritization which are considered major processes in regression testing. In this section only the related research to test suite prioritization is presented. Prioritization techniques can be based on many perspectives and factors such as risk, cost, and coverage criteria. Test cases prioritization are proposed based on software coverage techniques such as statement coverage, or branch coverage. In statement coverage prioritization technique each test case is executed and the number of statements is calculated and then the test cases are prioritized according to the number of covered statements. In branch coverage, each test case is executed and the number of covered branches is calculated. Test cases are then prioritize according to the number of covered branches. Mutation testing is used as well in prioritization through assessing and determining the number of mutants that have been killed by every test case and the test case that killed the highest number of mutants is considered to be the highest from prioritization perspective [12][13] . De Souza et al. present an approach in test case selection using particle swarm optimization and based on two objectives. The first objective is the execution cost and the second is the functional requirement coverage [7]. He et al. propose an approach in test suite reduction through combining information from both execution cost and code coverage using genetic algorithms [8]. Yoon et al. propose a technique for new test cases' prioritization through evaluating the value of risk disclosure for requirements along with the risk analyst to assess the related test cases and in that way the priority of test cases is determined [9]. Stallbaum, Metzger and Pohl introduce an approach in test case prioritization called RiteDAPin [10]. This approach creates automatically test case priorities through evaluating activity diagram risk information. RiteDAPinserts new risks to the activity diagram according to the associated reaction. The test case scenario priority is decided according to the reactions which have the maximum total of risk values. Chen, Probert and Sim prioritize the test cases based on the safety test through assessing severity probability and cost of test cases. The severity probability is computed through multiplying the defects number by the average severity of defects [11].

III. 3. Textual Similarity Based Test Suit Reduction

Similarity between test cases and methods (STcM)(tci,mj) is the cosine among vector tci and vector mj

Proc. of The Fifth Intl. Conf. On Advances in Computing, Electronics and Electrical Technology - CEET 2016
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-087-3 doi: 10.15224/ 978-1-63248-087-3-65

following indexing. While the range values of cosine is from -1 to 1. Thus when the value is close to 1 it means that the test case is more similar to the compared method.

$$\boldsymbol{StcM(tci, mj)} = \frac{\boldsymbol{vtci^T\ vmj}}{\boldsymbol{|vtci|_2\ X\ |vmj|_2}} \qquad (1)$$

Total weight for a given test case (TWTCi) is the summation of the similarity between all the methods in the system under test for the given test case.

$$TWTCi = \sum_{n=0}^{j} StcM(tci, mn) \qquad (2)$$

Total weight for a given test suite (TWTSi) is the summation of all total weightsof test cases in the test suite:

$$TWTSi = \sum_{n=0}^{j} TWTCi \qquad (3)$$

IV. Experiments and Results

With the intention of assessing test suite prioritization using similarity weight, an open source code is selected. The open source code is called MARC4J, which is an application for dealing with MARC records [14] . The reason behind selecting this application is that it includes a test suite that is generated and constructed by an independent party, and thus no bias is occurred.

The experiment consists of the following stages: first the similarity between each test case and method in MARC4J application is calculated as shown in Table 1. After that the total weight for a given test case is calculated and then the total weight for the whole test suite is calculated as shown in Table2. Finally, the coverage is calculated for each test suite using EclEmma to evaluate the proposed approach. EclEmma is used to provide coverage property for six different coverage counters: Instruction, Branches, Lines, Methods, Types and Complexity as shown in Figure 1 and Table 3. EclEmma is an Eclipse open source coverage tool [15] .

TABLE I: AN EXAMPLE OF SIMILARITY AMONG TEST CASES AND METHODS

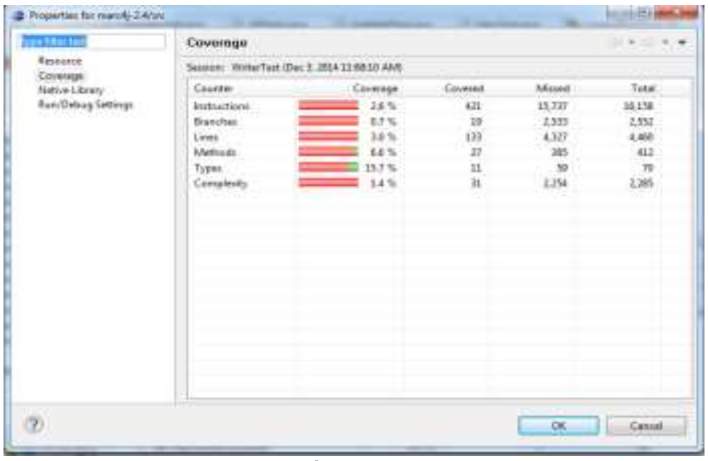
Test Case Name	Method Name	Class Name	Weight
testFind	Find	ControlFieldImpl	0.745
	getData	ControlFieldImpl	0.167
	Find	DataFieldImpl	0.745
	Find	RecordImpl	0.745
	Find	RecordImpl	0.745
	Find	RecordImpl	0.745
	Find	SubfieldImpl	0.745
	getData	SubfieldImpl	0.167
	getData	ControlField	0.167
	DataField	DataField	0.2
	Record	Record	0.187
	Find	Record	0.745
	Find	Record	0.745
	Find	Record	0.745
	Subfield	Subfield	0.071

Table 1 shows the similarity(weight) among the test case "testFind" with methods in MARC4J application. The results are calculated through employing LSI technique. The highest weight means that the test case and method are more similar in comparison with low weight values. In other words, the body of the test case and the body of the method is more likely when the weight is high.

Table II Test Suite Weight

Test Suite	Weight
DataFiledTest	3.6
ControlFieldTest	0.4
RecordTest	11.3
LeaderTest	3.9
ReaderTest	11.5
WriterTest	31.5
RoundtripTest	12.5

Table 2 shows the weight for each selected test suites. The results show that the WriterTest has the highest weight. This means that the test cases in the WriterTest test suite will cover more code than other test cases in other test suites. In addition the results show that the RoundtripTest ,ReaderTest and Record test have almost similar weights, and the LeaderTest and DataFieldTest have also similar weights where ControlFieldTest has the lowest weight .



Coverage

Session: WriterTest (Dec 3, 2014 11:08:10 AM)

Counter	Coverage	Covered	Missed	Total
Instructions	2.6 %	421	15,737	16,158
Branches	0.7 %	19	2,533	2,552
Lines	3.0 %	133	4,327	4,460
Methods	6.6 %	27	385	412
Types	15.7 %	11	59	70
Complexity	1.4 %	31	2,254	2,285

Figure1. EclEmma output

Figure1 shows the coverage for the WriterTest suite. The results show that the Type, Methods, Lines, Instructions, Complexity, and Branches coverage counters obtained are: 15.7%, 6.6%, 3%, 2.6%, 1.4% and 0.7% respectively. In addition, the results show that this test suite has achieved the highest coverage in comparison with all other test suites. The results prove that the test suites with high weight covers more than the test suite with low weight.

SEEK DIGITAL LIBRARY

Proc. of The Fifth Intl. Conf. On Advances in Computing, Electronics and Electrical Technology - CEET 2016
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-087-3 doi: 10.15224/ 978-1-63248-087-3-65

Table 3: Test Suite coverage Results

Test Suite	Instruction	Branches	Lines	Methods	Types	Complexity	Total
ControlFieldTest	0.7	0.4	0.9	3.4	5.7	0.7	11.8
DataFiledTest	1	0.5	1.4	5.6	7.1	1.1	16.7
LeaderTest	2	0.7	2.1	8	4.3	1.6	18.7
ReaderTest	1.7	0.6	2	7.9	12.9	1.1	26.2
RecordTest	0.6	0.4	0.8	1.9	5.7	0.5	9.9
RoundtripTest	1.1	0.4	1.4	3.6	10	0.8	17.3
WriterTest	2.6	0.7	3.0	6.6	15.7	1.4	30

Table 3 shows the results of the experiments. The results show that the test suite "WriterTest" has achieved the highest coverage (30%) in comparison with other test suites. Then the "ReaderTest" comes the second with (26.2%). "DataFiledTest", "LeaderTest" and "RoundtripTest" have achieved (16.7%, 18.7%, 17.3%) respectively. Moreover the "RecordTest" achieved the lowest coverage(9.9%). Whereas the weight of "RecordTest" is higher than the weight of "ControllFiledTest", this is because "RecordTest" consists of just two test cases and their implementations involves common methods that are used extensively in the source code. The proposed approach might be helpful in the classification of test suites into other levels according to their usage(e.g. generic or specific) .

V. Conclusion

In this paper a new technique for test suite prioritization is presented. The technique is based on the textual similarity among test cases in the test suite and the methods of the system under test. The approach is different than other approaches from thepre-knowledge of the test execution results. The proposed approach does not require any pre-knowledge or results of test cases execution. The textual similarity is determined and calculated through using latent semantic indexing technique (LSI). The results of the experiment show the effectiveness of the proposed technique.

References

[1] Huang, LiGuo, and Barry Boehm. "How much software quality investment is enough: A value-based approach." Software, IEEE 23.5 (2006): 88-95.

[2] Kit, Edward. Software testing in the real world. Addison-wesley, 1995.

[3] Whittaker, James A. "What is software testing? And why is it so hard?."Software, IEEE 17.1 (2000): 70-79.

[4] Rothermel, Gregg, et al. "Prioritizing test cases for regression testing."Software Engineering, IEEE Transactions on 27.10 (2001): 929-948.

[5] Krishnamoorthi, R., and S. A. Sahaaya Arul Mary. "Factor oriented requirement coverage based system test case prioritization of new and regression test cases." Information and Software Technology 51.4 (2009): 799-808.

[6] Yoo, Shin, and Mark Harman. "Regression testing minimization, selection and prioritization: a survey." Software Testing, Verification and Reliability 22.2 (2012): 67-120.

[7] de Souza, Luciano S., et al. "A multi-objective particle swarm optimization for test case selection based on functional requirements coverage and execution effort." Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on. IEEE, 2011.

[8] He, Zhen-feng, Bin-kui Sheng, and Cheng-qingYe. "A genetic algorithm for test-suite reduction." Systems, Man and Cybernetics, 2005 IEEE International Conference on. Vol. 1. IEEE, 2005.

[9] Yoon, Miso, et al. "A Test Case Prioritization through Correlation of Requirement and Risk." Journal of Software Engineering and Applications 5.10 (2012): 823.

[10] Stallbaum, Heiko, Andreas Metzger, and Klaus Pohl. "An automated technique for risk-based test case generation and prioritization." Proceedings of the 3rd international workshop on Automation of software test. ACM, 2008.

[11] Chen, Yanping, Robert L. Probert, and D. Paul Sims. "Specification-based regression test selection with risk analysis." Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research. IBM Press, 2002.

[12] Rothermel, Gregg, et al. "Prioritizing test cases for regression testing."Software Engineering, IEEE Transactions on 27.10 (2001): 929-948.

[13] Andrews, James H., et al. "Using mutation analysis for assessing and comparing testing coverage criteria." Software Engineering, IEEE Transactions on 32.8 (2006): 608-624.

[14] http://marc4j.tigris.org/

[15] http://www.eclemma.org/

About Author:

Iyad M Alazzam, is an assistant professor in the department of computer information systems at Yarmouk University in Jordan, he has received his Ph.D degree in software engineering from NDSU (USA). His master from LMU (UK) in electronic Commerce and his B.Sc in computer science and information systems from Jordan University of Science and Technology in Jordan. His research interests lays in software engineering and software testing.