TEST CASE PRIORITIZATION METHODLOGY FOR REAL TIME SOFTWARE DEVELOPMENT

Surendra Mahajan[1], Dr. S. D. Joshi[2], Dr. V. Khanna[3]

[1] *Research Scholar, Department Of Computer Science and Engineering, Bharath University, (India)*

[2] *Department of Computer Engineering, Bharathi Vidyapeeth Deemed University, (India)*

[3] *Department Of Computer Science Engineering, Bharath University, (India)*

ABSTRACT

Software testing and retesting happens persistently amid the software development lifecycle to discover lapses as ahead of schedule as could be allowed. The sizes of test suites develop as software develops. Because of asset obligations, it is vital to prioritize the execution of test cases to expand possibilities of ahead of schedule recognition of shortcomings. Earlier procedures for test case prioritization are focused around the aggregate number of scope prerequisites practiced by the test cases. In this paper, we introduce another methodology to prioritize test cases focused around the scope prerequisites exhibit in the applicable areas of the domain of test cases.

Keywords: Heuristic, Regression Testing, Software Testing, Test Case, Test Case Prioritization

I INTRODUCTION

Software testing is a vital and lavish phase of software development. As software changes about whether, test suites are created and used to test the changed software to verify that changes don't influence the current usefulness in unintended routes, and to test for new usefulness. This methodology is called regression testing. Because of time and asset requirements, it may not be conceivable to constantly execute all the tests in suites on every testing emphasis. It is in this manner essential to prioritize the execution of test cases in test suites to execute those test cases at an opportune time amid regression testing, whose yield is more inclined to change. Such a prioritization is required to help in right on time location of flaws amid regression testing. Strategies for Test Case Prioritization address this issue. In this paper, we show another methodology for prioritizing the execution of existing test cases with the objective of ahead of schedule recognition of shortcomings in the regression testing procedure.

The former test case prioritization strategies mulled over in [1,2,3,4] are principally focused around varieties of the aggregate necessity scope and the extra prerequisite scope of different structural components in a system. For example, absolute proclamation scope prioritization requests test cases in diminishing request of the amount of explanations they work out. Extra proclamation scope prioritization requests test cases in diminishing request of the amount of extra explanations they work out, that have not yet been secured by the tests prior in the prioritized succession. These prioritization methods don't think seriously about the announcements or limbs that really affected,

or could possibly impact, the estimations of the project yield. Not, one or the other do they mull over whether a test case navigates a changed articulation or not while prioritizing the test cases.

II LITERATURE REVIEW

It is natural to expect that the yield of a test case, that executes a bigger number of articulations that really impact the yield or can possibly impact the yield, is more inclined to get influenced by the alteration than tests coating less such explanations. Furthermore, tests practicing adjusted proclamations ought to have higher necessity than tests that don't cross any alterations. In this paper, we introduce another methodology for prioritizing test cases that is built not just with respect to aggregate extension scope; however that likewise considers the amount of articulations (limbs) executed that impact or can possibly impact the yield created by the test case. The set of articulations that impact, or can possibly impact, the yield of a project when run on a specific test case, relate to the applicable cut [5,6,7] processed on the yield of the system when executed by the test case. Our methodology is focused around the accompanying perception. In the event that an adjustment in the project need to influence the yield of a test case in the regression test suite, it must influence some calculation in the pertinent cut of the yield for that test case. In this way, our heuristic for prioritizing test cases relegates higher weight to a test case with bigger number of articulations in its pertinent areas of the yield [8,9,10].

Our methodologies show changes over former methodologies focused around aggregate extension scope and give fascinating experiences into the utilization of size of significant cuts of yields in deciding the relative vitality of test cases in uncovering a change in the yield amid regression testing.

There are many issue of test case prioritization exists and those are enhancement issues as well as have exponential unpredictability in the most detrimental possibility. In [8,9], the introduced procedures are really avaricious heuristics in which test cases are requested based upon their necessity scope. In their exploratory studies in [6,11,12], the projects were utilized to assess the adequacy of the above heuristics. Each one project is connected with a suite of defective adaptations and a pool of test cases. Consequently, the viability of the above test case prioritization heuristics in ahead of schedule execution of test cases that influence the yield was measured by figuring the rate at which the seeded flaws were uncovered by the test cases in the prioritized grouping. This rate was quantitatively measured by registering the APFD values for the prioritized test suites.

III RESEARCH METHODOLOGY

Instinctively, if a test case practices a changed explanation, and that altered proclamation is additionally included in the significant cut of the yield, then in all probability the yield of the test case will be influenced because of the adjustment. This is on the grounds that each announcement in the applicable either influences the yield or can possibly influence the yield of the test case.

Then again, there might be adjustments, for example, if a worth is increased and therefore decremented in the code before the quality is utilized again as a part of which the change may not influence the yield. We outline this with the illustration in algorithmic steps as follows:

1: readTest(a, b, c); // a,b,c are test cases

2: int x=0, y=0, z=0;

3: x := a + 1; // x,y,z are test suites

4: y := b + 1;

5: z := c + 1;

6: int w := 0; // w is prioritization test suite

B1: if x > 3 then

B2: if z > 4 then

7: w := w + 1;

endif

endif

B3: if y > 5 then

8: w := w + 1;

endif

9: write(w);

Let us consider the case in which a modified statement is not contained in the relevant slice. A test case exercising a modified statement, that is not contained in the relevant slice, will likely change the program output only when the variable being defined by the modified statement is changed to another variable which is used in the relevant slice. However, we believe this case to be the only case in which an exercised modification will be outside the relevant slice, yet may still affect the program output.

IV ANALYSIS

We used the News Blaster programs as described in Table 1. Each program is associated with a set of faulty versions and a pool of test cases.

TABLE 1. NEWS BLASTER TEST PROGRAMS

Program Name	Lines of Code	Fault Level (%)	Test Case Size
Sentence Processor	120	80	1520
History Search	98	43	1034
Save History	150	9	2356
Lookup Table	212	12	4211
Connection Pooling	70	65	1412
Active Directory	205	76	5345

The objective of our trials is to perceive how well our prioritized suites for the heuristic proposed in this paper perform as far as rate of flaw identification, concerning the broken adaptations gave the News Blaster Test Programs.

Our trial methodology is to create test suites, prioritize them, and afterward measure the ensuing APFD esteem for each one prioritized suite. Like the trial setup in, we created 100 extension scope sufficient test suites from the given test pools to each one system. For each one test case, we measured the aggregate set of practiced articulations and extensions utilizing instrumented renditions of the subject projects as produced by the Aristotle program dissection instrument [6]. For each one test case we likewise registered the set of articulations and extensions in the significant cut focused around the project yield produced when executing the test case on the given subject system. To figure an important cut, we initially processed the announcements in the element cut of the project yield by figuring the transitive conclusion of information and control conditions of the yield practiced by the test case. At that point, we enlarged the element cut with the extra articulations/branches that were conceivably yield impacting (this would regularly include static investigation, yet we attempted to rough this by taking a gander at the execution hints of every last one of tests in the substantial test pools furnished with the Siemens suite projects to check whether changing the limb result would influence the project yield or not) and their relating information conditions to acquire the significant cuts.

Given the sets of standard practiced articulations and extensions alongside the relating important cuts for each one test case, we utilized this data to prioritize the test cases and a while later figured the APFD esteem for each one prioritized suite to assess the suite's rate of fault detection. We analyzed the sorts of errors presented in the broken variants of each one subject program and distinguished six unique classes of seeded errors: (1) changing the

administrator in a statement, (2) changing an operand in a representation, (3) changing the estimation of a consistent, (4) uprooting code, (5) including code, and (6) changing the sensible conduct of the code (typically including a couple of alternate classifications of error sorts all the while in one flawed variant). Therefore, the defective variants utilized as a part of our investigations blanket a wide mixed bag of issue sorts.

V CONCLUSION

While we may expect that a practiced alteration that is likewise included in the important area will influence the yield of a project, there do exist cases in which practicing a change, that is held in the applicable area, will really not change the yield value(s) of a system and accordingly may not prompt a deficiency being uncovered.

This paper intimates that there is no assurance that a test case practicing more changed explanations that are held in the significant area, will essentially expose more faults than an alternate test case practicing less alterations that are in the important area. Also, in practice we may expect a higher probability that practicing an adjustment that can possibly influence project yield, will really cause system yield to change and hence uncover a flaw. This system definitely focuses over the practical approach to resolve real time software development difficulties

REFERENCES

[1] Zhang, Lingming, et al. "Bridging the gap between the total and additional test-case prioritization strategies." *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, 2013,pp.73-79

[2] Thomas, Stephen W., et al. "Static test case prioritization using topic models." *Empirical Software Engineering* 19.1 (2014), pp.182-212.

[3] Arafeen, Md, and Hyunsook Do. "Test Case Prioritization Using Requirements-Based Clustering." *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 24-28.

[4] Lin, Chu-Ti, et al. "History-based Test Case Prioritization with Software Version Awareness." *Engineering of Complex Computer Systems (ICECCS), 2013 18th International Conference on*. IEEE, 2013, pp-192-196.

[5] Do, Hyunsook, et al. "The effects of time constraints on test case prioritization: A series of controlled experiments." *Software Engineering, IEEE Transactions* 2010, pp. 593-617.

[6] Singh, Yogesh, Arvinder Kaur, and Bharti Suri. "Test case prioritization using ant colony optimization." *ACM SIGSOFT Software Engineering* 2010,pp.110-117.

[7] Zhai, Ke, et al. "Taking advantage of service selection: a study on the testing of location-based web services through test case prioritization." *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE, 2010, pp. 450-456.

[8] Chen, Lin, et al. "Test case prioritization for web service regression testing." *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*. IEEE, 2010, pp. 67-69.

[9]Li, Sihan, et al. "A simulation study on some search algorithms for regression test case prioritization." *Quality Software (QSIC), 2010 10th International Conference on*. IEEE, 2010, pp.110-114.

[10] Carlson, Ryan, Hyunsook Do, and Anne Denton. "A clustering approach to improving test case prioritization: An industrial case study." *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*. IEEE, 2011, pp.230-234.

[11] Nguyen, Cu D., Alessandro Marchetto, and Paolo Tonella. "Test case prioritization for audit testing of evolving web services using information retrieval techniques." *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 78-81.

[12] Maia, Camila Loiola Brito, et al. "Automated test case prioritization with reactive GRASP." *Advances in Software Engineering* 2010, pp-14-18.