Time Based Prioritization of Test Cases in Regression Testing

Jatinder Singh*
House No. 9, Gali No. 21, Karan Vihar, Karnal, Haryana – 132001, India; jatindersinghswach@gmail.com

Abstract

Background/Objectives: When client wants some change in functionality or in look/design of software, then after implementation of requirement, tester has to retest it. For this tester has to perform regression testing. It is done to verify that any other issue is not pushed into the system, and also verify existing issue if any. So, Regression testing not only is very important when software change made but also it is very costly too. Methods: This paper proposes a new method based on ratio between number of issues found and run time of Test cases. This paper gives predilection to the original test suite so that the new suite, which is run within a time-constrained execution environment, will have a superior rate of fault detection. Findings: This paper presents a technique that prioritizes regression test cases so that the new ordered test case produces a high productivity. Application/Improvements: This paper assumes data for pre-executed test cases in the form of number of issues found and run time.

Keywords: How to Prioritize Test Cases Based on Time, Priority Techniques Basis of Time Constraints, Regression Testing, Test Suite Precedence Based on Time

1. Introduction

Regression test just means rerun previously run test cases for affected areas. This change can be due to Change in requirement of client or bug fixing or new feature added in software or Performance issue fix etc. For Regression testing tester has various options like: 1. Test all the scripts, or 2. Use some related test cases. First option is obviously not feasible due to constraints of resources like tester or time or system, so need to go for second one. But for this method tester have to rigorously select related scripts and categorize them in two categories e.g. Obsolete and Reusable. Then from reusable test cases will make priority and arrange them accordingly. In short, the process of regression test is as follows:

- Identify the existing test scripts that should be re-run,
- Create templates from test scripts,
- Select and group templates into sets,
- Select and deploy regression test sets, and
- Track our regression test results.

Now, Tester can do regression testing by running again these tests cases partially, i.e. not fully, to check for bugs which are fixed previously. Tester needs to write new test cases for changed scenario or test cases for new functionality wherever necessary. Test case prioritization is mainly used to improve regression testing by reordering the test cases so that the important test cases are run first some factors to consider during this process include the following:

- The issue is fixed but it is possible that due to this issue fixing another issue may be created or introduced.
- Regression test for every issue fixed must be written.
- Choose from the list of issues which are of similar types, next choose issue of high impedance and remove others.
- Needs to pay attention only on which are related with functionality only and ignore other UI issues.

*Author for correspondence
• Inspect the changes on program memory.

In test case prioritization, test suites i.e. group of test cases will be created to improve the testing process. This paper produced a more improved test case prioritization if the maximum allotted time is already known to each test case.

2. What is going on presently?

On the basis of studies, regression testing consumes approximately 50% of software system development resources².

3. Some Improvements through Automation

If automate testing is used then the total cost of developing software can be reduced drastically; some of benefits of automated testing are:

• The test case preparation can be done in advance;
• The test case execution would be significantly speedy; and
• The assurance of the testing result can be improved.

But as per studies happen so far, automation process of software testing is not an easy job to do. The main thing in this is to feed data in automated testing³. Which is not an easy task to find or generate?

4. Some more Improvement through AI in Testing

Artificial Techniques are used in software testing. It is still emerging and developing area. These concerns impose the need to investigate the excellence of search algorithms like genetic algorithms, simulated annealing and ant colony optimization as an enhanced substitute for developing test data generators⁴.

Through use of evolutionary computations, researchers have investigated genetic algorithms (GA)-based test data generators⁵. Few years ago, variety of techniques for test data generations have been already developed⁶ and these can be categorized as functional and structural testing. Full software testing is always not possible and not feasible as well, because it needs lots of resources like time and humans and money. Therefore, more selective approach should be developed to test by focus on the parts which are most critical⁷.

5. Building a Test Library

Most efficient regression testing can be done through developing a set or library of test cases. This library includes standard test cases which can be run each time when made some changes in code. After this apply test case library and will update these on the basis of changes made.

In test library, various types of issues can be found, for example, automated test cases, corner valued test cases and test cases which found issues previously.

After that, regularly, go through our test cases so that unrelated test cases can be removed i.e. the test cases which are not relevant now and have to remove it from test library. This will have to repeat this on regular period basis e.g. every 6 months etc. But whenever found any duplicate test cases then it need to retain the most important test cases which is capable to found maximum issues and other test cases can drop or remove from library.

6. Proposed Method

The proposed technique is to set priority of test cases on the basis of ratio between issues found and execution time. It can be understood through following example.

Example assumes that regression test suite (TS in short) contains nine test cases (TC<n> in short) with the initial order as follows: TC1, TC2, TC3, TC4, TC5, TC6, TC7, TC8, TC9 as shown in Table 1. This example also assumes to have data regarding faults detected by TS in the program P1 by each test case-wise.

| Test Cases | Issues  | Run Time (Mins.) |
|------------|---------|-----------------|
| TC1        | I2, I4, I5 | 6               |
| TC2        | I1, I2   | 2               |
| TC3        | I2, I3, I5 | 3               |
| TC4        | I1, I3, I4 | 6               |
| TC5        | I3, I4   | 1               |
| TC6        | I1       | 1               |

Table 1. Raw data for test cases and number of issues found with its run time.
As shown in Table 1 TC1 found issues # 2,4,5; TC2: 1,2; TC3:2,3,5; TC4:1,3,4; TC5: 3,4; TC6: 2,3,5; TC7: 1; TC8: 2 and TC9: 3.
Table 1 also shown that execution time of TC1 is 6 minutes; TC2 is 2 minutes; TC3 is 3 minutes, TC4 is 6 minutes, TC5 is 1 minute, TC6 is 1 minutes; TC7 is 0.5 minute; TC8 is 1.5 minutes and TC9 is 2 minutes.
For ease reformat the data as shown in Table 2.

| Test Cases | I1 | I2 | I3 | I4 | I5 | No. Issues | Run Time |
|------------|----|----|----|----|----|------------|----------|
| TC1        | N  | Y  | N  | Y  | Y  | 3          | 6        |
| TC2        | Y  | Y  | N  | N  | N  | 2          | 2        |
| TC3        | N  | Y  | Y  | N  | Y  | 2          | 3        |
| TC4        | Y  | N  | Y  | Y  | N  | 3          | 6        |
| TC5        | N  | N  | Y  | Y  | N  | 2          | 1        |
| TC6        | Y  | N  | N  | N  | N  | 1          | 1        |
| TC7        | N  | Y  | Y  | N  | Y  | 3          | 0.5      |
| TC8        | N  | Y  | N  | N  | N  | 1          | 1.5      |
| TC9        | N  | N  | Y  | N  | N  | 1          | 2        |

Now it can be seen the performance of each test case in Table 3.

| Test Cases | No. of Issues | Run Time (minutes) | Average Issue/Min. |
|------------|---------------|--------------------|---------------------|
| TC1        | 3             | 6                  | 0.5                 |
| TC2        | 2             | 2                  | 1.0                 |
| TC3        | 2             | 3                  | 0.6                 |
| TC4        | 3             | 6                  | 0.5                 |
| TC5        | 2             | 1                  | 2.0                 |
| TC6        | 1             | 1                  | 1.0                 |
| **TC7**    | **3**         | **0.5**            | **6.0**             |
| **TC8**    | **1**         | **1.5**            | **0.6**             |

TC1 found 3 issues in 6 minutes, so the average between No. of issues (3) with run time (6) is 3/6=0.5, will have following averages for test cases are: TC2: 1.0; TC3: 0.6; TC4: 0.5; TC5: 2.0; TC6: 1.0; TC7: 6.0; TC8: 0.6; TC9: 0.5;
In Table 3 it is very much clear that TC7 have the highest average ratio between no. of issue found and run time i.e. 6.0. So choose TC7, TC5, TC2, TC6 to be retained in Test case library and rest of issues can be dropped off.

7. Conclusion

This paper proposes a new technique to choose test cases from large pool of test cases. This technique reduces selection time drastically and much helpful to perform regression testing in tight timeline of schedule and improves the quality of software testing.

8. References

1. Krishnamoorthi R, Mary SA. Regression Test Suit Prioritization using Genetic Algorithms, International Journal of Hybrid Information Technology. 2009; 2(3):35–52.
2. Aditya PM. Foundation of Software Testing. 1st Edition, Pearson Education; 2008.
3. Ranjan SP. Generation of Test Data using Meta Heuristic Approach IEEE TENCON. 2008 Nov.
4. Ranjan SP, Tai-Hoon K. Application of Genetic Algorithm in Software Testing, International Journal of Software Engineering and its Applications. 2009; 3(4):1−10.
5. Nashat M, Miran S. Data Generation for Path Testing, Software Quality Journal. Kluwer Academic Publishers. 2004 Jun; 12(2):121–36.
6. Wegener J, Baresel A, Sthamer H. Suitability of Evolutionary Algorithms for Evolutionary Testing. In: Proceedings of the 26th Annual International Computer Software and Applications Conference; 2002 Aug. p. 26–29. https://doi.org/10.1109/CMS-PSAC.2002.104566.
7. Lin JC, Yeh PL. Using Genetic Algorithms for Test Case Generation in Path Testing. In: Proceedings of the 9th Asian Test Symposium (ATS’00); Dec 2000.