IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 8, August 2016

Iterative Pareto Principle for Software Test Case Prioritization

Manas Kumar Yogi[1], G. Vijay Kumar[2], D. Uma[3]

Assistant Professor, CSE Department, Pragati Engineering College, Surampalem, India [1, 2, 3]

Abstract: This paper introduces a repetitive nature of application of Pareto principle for test case prioritization. In software testing, situations demand effective resource and time utilization which forces the testing team to execute the test cases which uncover more number of bugs. So, taking the Pareto Principle we apply the 80-20 rule in a repeated manner until a satisfiable state is reached. The proposed mechanism precisely does what it is modelled for, test case prioritization.

Keywords: Pareto Principle, Testing, Prioritization, Bugs, Iterative

I. INTRODUCTION

Most applications today have been designed to include some form of customization or extensibility including user preference settings, scripting languages or APIs for custom extensions using more traditional languages.
Software companies like Microsoft understand this, and design for a product that addresses 80 percent of the requirements, leaving the last 20% as customizations by the end user. 80% of Defects are caused by 20% of Code
The concept here is the Pareto Principle, originally described by Vilfredo Pareto and later formalized by Joseph Juran. Of course, this is just a rule of thumb, but an important one. Whether the percentages are really 70/30 or 90/10, the reality is that most things are caused by a few underlying factors. For software testers, knowing this fact can offer tremendous value. If a tester is simply looking at a list of 100 bugs, it may not be clear if there is any underlying meaning. But if the tester were to combine those bugs based on some kind of category, it may be possible to see that a very large number of bugs come from very few places.
Here are a few recommendations for getting the most out of this principle:

- Try to sort bugs by root cause and not by outcome. Grouping all the bugs that made the software crash isn't that helpful. Grouping all the bugs that resulted from module XYZ is more helpful.
- Work with developers to look for innovative groupings. For example, 80% of the program's bugs may result from calling the same underlying library. However, that may not be readily apparent from where the bugs occur within the program.
- Remember that bugs may result from flawed procedures. For software testers, knowing this fact can offer tremendous value. If a tester is simply looking at a list of 100 bugs, it may not be clear if there is any underlying meaning. But if the tester were to combine those bugs based on some kind of category, it may be possible to see that a very large number of bugs come from very few places.

II.PROPOSED MECHANISM

For our proposed mechanism, we take a suite of test cases ,say test cases numbered from T1 to T10.Each test case is tabulated as below with ability to uncover the specified number of bugs. We have arranged the test cases in decreasing order of the fault finding ability based on number of bugs they have uncovered.

TABLE I

Test Case	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
No. of Bugs found	10	9	8	7	6	5	4	3	2	1

We propose to apply the Pareto principle ie 80-20 rule to this test cases. We advocate the principle as 80% of the bugs are found by only 20% of the test cases.

In our case, out of 55 bugs found by the 10 test cases,80% of 55 ie, 44 bugs are found by the test cases,T1,T2,T3,T4,T5,T7.So,we have found the test cases which are to be executed first when testing time is limited. We now extend this principle iteratively till we get only the test case which has the ability to uncover maximum bugs.

We have presented in below table the iterations after applying Pareto Principle to the data obtained from Table I. In each iteration, the test cases which sum up to the total of 80% of the bug count are mentioned in the same row. For example, in first iteration, 80% of 55 test cases (T_1+T_2+..+T_{10}) is 44 which can be detected by 6 test cases i.e, T_1, T2, T3, T4, T5, T7 out of the total number of 10 test cases. In iteration 2 similarly, 80% of 44 =35 which needs test cases T_1, T2, T3, T4, and T10 to execute. Likewise we apply Pareto rule until we reach iteration 7 where only a single test case T_1 is obtained. We terminate the iteration at this juncture as further down we don't get even single test case. Evidently, at least one test case must be executed.

TABLE III

Iteration Number	80% OF TEST CASES	Test cases
1	44	T_1,T_2,T_3,T_4,T_5,T_7
2	35	T_1,T_2,T_3,T_4,T_{10}
3	28	T_1,T_2,T_3,T_{10}
4	22	T_1,T_2,T_8
5	17	T_1,T_4
6	13	T_1,T_8
7	10	T_1

Now, our aim is to construct a test cases execution order with respect to the table above. This order will be the prioritised test cases order for final execution of test cases. For this, we scan the table from up to down and pick out the test case which occurs maximum number of times in each iteration.
In our example, it's T1, which occurs 7 times. So, in prioritised order T1 appears first .Then,T2 appears 4 times,T3,T4 appear 3 times each, then T10,T8 appears 2 times and finally T5 appears once. We can take a note that, T6 appears 0 times in the table.Subsequently, T6 is to be given least preference when we construct the test case prioritization order. Hence we obtain the final test cases execution order as stated below:
{T1, T2, T3, T4, T10, T8, T5, T7, T6}.

III.THREATS TO VALIDITY

Following are the threats to the validity of our model:

1. The nature of bugs are not considered, ie more than 2 test cases may find same nature of bug. We have assumed only count of bugs for each test case.

2.We have stopped the application of Pareto principle once the test case with highest count of bugs reaches 80% of cycle i where i=1 to n. In our example, n=7.Beyond this also we could proceed but our objective will not be strictly satisfied.

3. Our model will behave in same way for substantial amount of test cases say, when test cases are in 1000, we could not experiment on that. That is the future work for our proposed model.

IV.CONCLUSION

Pareto principle helps us in giving a direction towards application test case prioritisation. When the testing team has many test cases in hand then our mechanism can be used to order the test cases giving priority to certain test cases and leaving the rest. It saves time as well as testing effort to a extent. How much time ,effort is saved that is the scope of our future work and we would like to extend this model to regression test suites as well .Our paper aims to assign a specific test case execution order using iterative action on test cases with the principle that only few of test cases have ability to determine large set of bugs.

REFERENCES

[1] Edward L. Jones ―Grading student programs – a software testing‖, Proceedings of the fourteenth annual Consortium for Computing Sciences in Colleges, 2000.
[2] Miller, William E. Howden, "Tutorial, software testing & validation techniques", IEEE Computer Society Press,1981.
[3] Ian Somerville, ‖ Software Engineering ‖, Addison-Wesley,2001.
[4] James Bach, ―Exploratory Testing Explained ‖, v.1.3 ,4/16/03.
[5] John E. Bentley, Wachovia Bank, Charlotte NC, ―Software Testing Fundamentals—Concepts, Roles, and Terminology ‖, SUGI 30.
[6] Myers, Glenford J., ―The art of software testing ‖, New York: Wiley, c1979. ISBN: 0471043281.
[7] Nick Jenkins. ―A Software Testing Primer ‖, 2008.
[8] Peter Sestoft, ‖ Systematic software testing ‖, Version 2,2008-02-25.