Volume 6, Issue 4, April 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Optimal Release Planning and Reliability Modelling of Multi-Release Software

Tanuja S. Mulla, Amol K. Kadam, Dr. S. D. Joshi
Department of Computer Engineering, Bharati Vidyapeeth Deemed University College of Engineering
Pune, Maharashtra, India

Abstract-- The model identifies the faults that were left in the software when it is in working phase while testing of the new code. Since the proposed software product is dependent only on time, it can be classified under one-dimensional modelling outline. But we also have to consider other factors like available resources, testing coverage etc. simultaneously. Thus considering those factors using a Cobb Douglas production function extending our own modelling framework a two-dimensional software reliability growth model for multi release have been developed. Here, we work out on an optimal release planning problem of the software which minimises the cost of testing of the software release that is to be brought into the market under the constraint of remaining a desired proportion of faults from the current release and here we used a genetic algorithm to solve the problem.

Keywords-- Software Reliability, Software Reliability Growth Model (SRGM), Multi-Release, Two dimensional.

I. INTRODUCTION

As with the world becoming a global village, and technology reaching the remotest corner of each nation, competition has opened up like never before. So, as new projects are launched into the market, a strategy to meet the challenges from competitors is needed. Software development is a time consuming, costly process with high quality targets.

As an illustration, suppose a firm develops antivirus software. Such a firm can begin with releasing the product that detects and removes viruses and spyware from the computer system. In their second release, they can provide the feature of defensive the system from virus infected emails. Next, they can add the trait of blocking spyware automatically fetch to next release. Finally, the fourth release can provide root kit protection along with removing hidden threats in the operating systems. Software products are not static, and each release has a limited lifecycle area.

Hence, there is an utmost need to model multiple releases in the software development process. In this paper, we develop a modelling framework for such multiple releases and its failure process.

We formulate an optimal release planning problem under the necessity of removing a desired proportion of faults from the current release which minimizes the cost of testing the software release to be brought into market.

II. LITERATURE REVIEW

They projected the log logistic reliability growth model, which can capture the increasing decreasing nature of the failure occurrence rate per fault. Equations to estimation the parameters of the existing finite failure NHPP models, as well as the log-logistic Model, based on failure data composed in the form of inter failure times are developed. There is also an existing analysis of two data sets, where the underlying failure process could not be adequately described by the existing models, which inspired the development of the log-logistic model. Thus, for modelling the reliability growth of software with multiple releases, based on this idea, this paper proposes a mathematical modelling framework for multiple releases of software products. The proposed model takes into consideration the combined effect of schedule pressure and resource limitations using a Cobb Douglas production function in modelling the failure process using a software reliability growth model. The model developed is validated on a four release failure data set. Another major concern for the software development firms is to plan their lease of the upgraded version. When different versions of the software are to be released then the firm plans the release on the basis of testing progress of the new code, as well as the bugs reported during the operational phase of the previous version. So, they formulated an optimal release planning problem which minimizes the cost of testing of the release that is to be brought into market under the constraint of removing a desired proportion of faults from the current release.

III. SOFTWARE RELIABILITY GROWTH MODEL DATA

There are two types of data for software reliability growth models. The first is the time at which the defect was discovered, and the second is number of defects discovered.

Test Time Data:

For a software reliability growth model developed during QA, the measure of time must relate to the testing effort. There are three possible types for measuring test time a. Calendar time, b. number of tests run c.execution (CPU) time.

Test time can be a calendar time, in particular if there is a devoted test group that continuously runs the test machines. However, the test effort is an often asynchronous, so number of tests run or execution time is commonly used in place of calendar time. There are some test-suites that execute 100 tests in an hour and other takes 24 hours to execute for sophisticated tests. The Software is under Stress when longer test cases run and thus have a higher probability of finding a defect. Thus, they have developed software reliability growth models using calendar time, number of tests, and execution time as a measure of time. The result shows that execution time is the best measure of test time.

Defect Data:

Potential defect discoveries are recorded as Tandem Problem Reports (TPRs). TPRs are analysed by software developers to determine if the new defect has been detected. TPRs are not always defects because they may represent a non-defect and because many other people or groups of people may discover the same defect. Non-defect TPRs may create confusion about how to use a set of software or what the software is supposed to produce. Defects that have been found by multiples of people or groups of people are usually called duplicates or rediscoveries. Duplicates are not included in the defect counts since the original defect report is counted. TPRs that do not show new defects are called "smoke" by software developers. The amount of smoke generated during Quality Analysis test varies over time and by release. The amount of smoke after the software is delivered to customers is usually higher since much type of customers may encounter the same failure. TPRs are categorized as the severity of the defect. The severities ranges from 3 (most severe) to 0 (least severe). The severity levels are assigned depending on how urgently the customer needs a solution.

Customer Impact No Impact: Can tolerate the situation indefinitely. Minor: Can tolerate the situation, but expect solution eventually. Major: Can tolerate the situation, but not for long. Solution needed. Critical: Intolerable situation. Solution urgently needed.

IV. PARAMETER ESTIMATION

A software reliability model is a function such as that fitting function to the data means evaluating its parameters from the data. One approach for estimating parameters is to fed up the data into equations directly for the parameters. The most commonly used method for this direct parameter estimation is the maximum likelihood technique. A second approach for estimating parameters is fitting the curve described by the function to the data and estimating the parameters from the best fit to the curve. The most commonly used method for the indirect parameter estimation is the least squares technique. The alternative least squares technique is used most often since it provides the best results. We propose a two dimensional software reliability growth model which models the number of faults removed by combining it with the effect of testing time and testing effort with the testing resources. The model in our study is formulated using Cobb Douglas production function which shows the effect on number of faults removed with respect to the testing time and testing effort of the software.

V. CONCLUSION

To capture the effect of faults generated in the software, we have developed multi-release software reliability modelling framework. The model uniquely takes into account the faults of that release which is under the phase of testing (i.e. the release which is to be brought into market) and the faults left in the previous release.

REFERENCES

[1] Swapna S. Gokhale1_and Kishor S. Trivedi "Log-Logistic Software Reliability Growth Model"Center for Advanced Computing and Communication, by the National Science Foundation grant number EEC-9714965 and Charles Stark Draper Laboratory Grant # DL-H-505333.

[2] Harold A. Strieber "A Family of Software Reliability Growth Models "International Computer Software and Applications Conference (COMPSAC 2007) 0-7695-2870-8/07 $25.00 © 2007 IEEE.

[3] Sunil Kumar Khatri, Deepak Kumar, Asit Dwivedi, Nitika Mrinal “Software Reliability Growth Model with Testing Effort using Learning Function” IEEE Transactions on Software Engineering SE-11, pp. 1411-1423, 2005.

[4] Jing Zhao Hong-Wei Liu Gang Cui Xiao-Zong Yang, "A Software Reliability Growth Model from Testing to Operation "Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM’05) 1063-6773/05 $20.00 © 2005 IEEE.

[5] P. K. Kapur, H. Pham, *Fellow, IEEE*, Anu G. Aggarwal, and Gurjeet Kaur "Two Dimensional Multi-Release Software Reliability Modeling and Optimal Release Planning" IEEE Transactions on reliability, VOL. 61, NO. 3, September 2012.

[6] Alan Wood "Software Reliability Growth Models: Assumption vs. Reality"1071-9458/97 $10.00 *0* 1997 IEEE.

