Estimation of Software Reliability Growth Model Parameters

Claes Wohlin
Blekinge Institute of Technology, Sweden
E-mail: Claes.Wohlin@bth.se

Abstract

Software reliability growth models only become useful if it is possible to estimate their parameters. However, the parameter estimation is normally done numerically, and hence a substantial amount of data is all too often needed. This need is often not fulfilled by modern development processes, for example incremental development and agile methods. This position paper identifies three alternative ways of estimating the parameters in the models. Data from a case study of two software releases is used to illustrate how one parameter can be estimated from historical data. Thus, alternative ways of estimating the model parameters may be one way of making the models useful in modern development practices.

1. Introduction

The traditional way of predicting software reliability has since the 1970ies been the use of software reliability growth models. They were developed in a time when software was developed using a waterfall process model. This is inline with the fact that most software reliability growth models require a substantial amount of failure data to get any trustworthy estimate of the reliability. Software reliability growth models are normally described in the form of an equation with a number of parameters that need to be fitted to the failure data. A key problem is that the curve fitting often means that the parameters can only be estimated very late in testing and hence their industrial value for decision-making is limited. This is particularly the case when development is done, for example, using an incremental approach or other short turnaround approaches. A sufficient amount of failure data is simply not available.

The software reliability growth models have initially been developed for a quite different situation than today. Thus, it is not a surprise that they are not really fit for the challenges today unless the problems can be circumvented. This paper addresses some of the possibilities of addressing the problems with software reliability growth models by looking at ways of estimating the parameters in software reliability growth models before entering integration or system testing.

2. Three approaches

One possible way to address the problem with software reliability growth models needing a lot of data to make stable predictions is to estimate the model parameters by other means. Three different approaches exist:

- Historical data from previous similar situations, i.e. a software reliability growth model parameter value is used from a similar project or situation,
- In-project estimation, i.e. parameters are estimated using information from the current project,
- Combined approach, for example building a model for estimating a parameter from historical data and then feeding the model with current data.

As an example the Goel-Okumoto model [1] can be mentioned that includes two parameters. The Goel-Okumoto model is a simple non-homogeneous Poisson process (NHPP) model with the following mean value function: $\mu(t) = a(1 - \exp(-bt))$ where a is the total number of failures expected to be found as t goes towards infinity. $\mu(t)$ is the expected number of found failures at time t. Finally, b is possible to view as a testing efficiency parameter. A higher value means that more failures will be found per time unit.

When considering how to estimate the model parameters, a basic understanding and interpretation of the parameters is important. If relating the Goel-Okumoto model and its parameters to the three approaches above, it becomes clear that the total number of failures is highly dependent on the current project and hence the first approach above is less suitable. On the other hand, the first approach may very well be suitable, if we apply a similar test approach, to estimate the test efficiency parameter. This is further elaborated in an example in Section 3. At this stage, it is also worth stressing that even if we are only capable of estimating one parameter, it makes the curve fitting considerably easier.

Going back to the number of failures expected to be found, two possible methods have been identified for estimation before integration and system testing: complexity metrics and capture-recapture estimations. Most work so far has been directed towards complexity metrics [2], although models based on them have also been criticized [3]. Complexity metrics require that a model is built from previous projects or increments, and then fed with new input to produce an estimate of the total number of failures to be expected. Capture-recapture has mainly in software engineering been used in software inspections [4], but some attempts to apply in to testing have also been published [5]. We are currently planning a study on using capture-recapture for typical components in the system together with an industrial collaborative partner. The idea is to identify a typical component and have several testers testing the component to get an estimate of the remaining number of defects in the component. We have chosen to use a typical component, since the company is not prepared to have several testers on each component only for estimation purposes. We are also looking into different ways of using this information for one typical component to scale the estimate to the whole system.

3. Example

As an example of the possibilities with estimating from historical data, a summary of the main findings from [6] is provided. Failure data was available for two consecutive releases. The development and test methods were similar between the releases. For the first release, we got the following estimates at the end of testing: $a = 199$ and $b = 0.0981$. This estimate of b is now used for the following release. The estimation of a is now straightforward and can be done as soon as failure data is available. In this particular case, the estimate did not become stable until week 14. However, when estimating both parameters for the second release the estimates did not become stable until week 21. The total test period was 28 weeks and hence it is a considerable advantage if we have a good estimate after 50% or the test period instead of after 75%. The estimate at week 14 was that $a = 270$, and at week 21 we got an estimate of $a = 277$ (based on only the current system). The estimate at the end of the test phase for the second release became: $a = 250$ and $b = 0.0999$. The success in this case is of course highly dependent on the fact that the b value turns out to be very similar. On the other hand, it shows that it is possible, and hence this case could be viewed as a "proof-of-concept".

4. Summary

Software reliability growth models started to be developed in an era where the waterfall model was king (or queen), but they are less useful in modern approaches to software development. Thus, we have either to invent completely new ways of capturing the information that is hidden in failure data or we have to adapt the usage of the software reliability growth models to current ways of developing software.

This position paper has pointed to some opportunities when it comes to applying software reliability growth models. Or more specifically to different ways of estimating the software reliability growth model parameters without having to wait until the solution of one or more non-linear equations can be solved numerically with a stable solution.

5. References

[1] A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software and Other Performance Measures" IEEE Transactions on Reliability, Vol. 28, No. 3, pp. 206-211, 1979.

[2] M. Zhao, C. Wohlin,, N. Ohlsson, M. Xie, "A Comparison between Software Design and Code Metrics for the Prediction of Software Fault Content", Information and Software Technology, Vol. 40, pp. 801–809, 1998.

[3] N. E. Fenton and M. Neil, "A Critique of Software Defect Prediction Models", IEEE Transactions on Software Engineering, Vol. 25, No. 5, pp. 675-689.

[4] H. Petersson, T. Thelin, P. Runeson and C. Wohlin, "Capture-Recapture in Software Inspections after 10 Years Research – Theory, Evaluation and Application", Journal of Software and Systems, Vol. 72, No. 2, pp. 249-264, 2004.

[5] C. Stringfellow, A. Andrews, C. Wohlin and H. Petersson, "Estimating the Number of Components with Defects Post-Release that Showed No Defects in Testing", Software Testing Verification and Reliability, Vol. 12, No. 2, pp. 93-122, 2002.

[6] M. Xie, G. Y. Hong and C. Wohlin, "Software Reliability Prediction Incorporating Information from a Similar Project", Journal of Software and Systems, Vol. 49, No. 1, pp. 43-48, 1999.