International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 4, ISSN (Online) 2249 –071X, ISSN (Print): 2278 – 4209

Selection of Software reliability Growth Model on the Basis of Real Time Data Sets

Tajinder Kaur
Asst. Prof., M.Tech. (IT)
SBBSIET College, Jalandhar
Email ID : kaurtajinder07@gmail.com

Jasjeet Kaur
M.Tech. (CSE)
SBBSIET College, Jalandhar
Email ID : jparmar21nice@gmail.com

***Abstract* — Software reliability is an important attribute for software quality. It is of prime importance for mission critical systems, which demand high reliability. Thus reliability modeling of software products and predicting reliability via such models at different phases of the software life cycle is of demand. To predict the reliability, a large number of software reliability models have been proposed in the literature. It proposed an algorithm for software reliability growth model selection. The proposed algorithm helps to analyze, manage and improve the reliability of the software products. With the help of correctly decided reliability objectives, one can determine when the software is good enough to release and also minimizes risks of deploying the software with serious failures.**

***Keywords* — Software Reliability, Model Selection, Release Decisions.**

I. INTRODUCTION

Software plays a critical role in our daily life since it is embedded in appliances, such as computers, automobiles and televisions etc., which are widely used. As software controls the entire system functions; the faults in the software may cause critical problems such as human death, injuries or financial loss. Thus it is important to use methods, which measure and control the reliability of the software in use. To measure and control the reliability, more than hundred models have been proposed in the literature (Lyu, 2007). But there does not exist a model that can be used in all cases and universally be recommended to users. It proposed an algorithm for software reliability growth model selection.

- *Software Reliability*

Software reliability is the probability that given software will be functioning without failure for a specified period of time in a specified environment. Software reliability is a key factor in software development process and software quality. Yamada et al. (1986, 1993), Huang and Kuo (2002), Musa (1999) proposed a SRGMs which described the explicit relationship amount the calendar testing time, the amount of test effort and the number of software errors detected by testing. The test effort is measured by the number of CPU hours, the number of executed test cases and so on.

Software reliability measurement and management in the Software development processes are essential to produce quality and reliable software efficiently and effectively. Quantitative measurement and management are characterizing the product reliability. In particular based on software-error data analyses, it is very important to evaluate software reliability during the software testing phase. Several software reliability models have been developed to describe a software-error detection phenomenon during the testing phase and to measure software reliability. Models which are concerned with the relationship between the time-interval between software failures at the time span of the software testing are called software reliability growth models. These models enable us to estimate software reliability measures such as the mean initial error content, the mean time interval between failures, the mean number of remaining errors at an arbitrary testing time point, and the software reliability function.

Most software reliability growth models have a parameter that relates to the total number of defects contained in a set of code. If we know this parameter and the current number of defects discovered, we know how many defects remain in the code. Knowing the number of residual defects helps us decide whether or not the code is ready to ship and how much more testing is required if we decide the code is not ready to ship. It gives us an estimate of the number of failures that our customers will encounter when operating the software. This estimate helps us to plan the appropriate levels of support that will be required for defect correction after the software has shipped and determine the cost of supporting the software.

- *Complexity of Software*

The complexity and size of a computer system have grown dramatically for the past two decades. Traditionally, most researches focused only on the design, improvement and reliability analysis of the hardware to reach the goal of high performance computer systems. But now, the growing trend of software criticality has generated more researches into the field of high-quality software development. In highly complex modern software systems, reliability is the most important factor since it quantifies software failures during the process of software development and software quality control. Software reliability is the probability that given software will be functioning without failure in a given environment during a specified period of time.

II. PROPOSED WORK FOR SOFTWARE RELIABILITY GROWTH MODEL SELECTION

- *AdaBoosting based Combinational Model*

Many SRGMs may result in estimation or prediction bias since their underlying assumptions are not consistent with the characteristics of the application data. To reduce this bias, combining several different SRGMs together in

linear or nonlinear manner is a common and applicable method. Many approaches are proposed to determine the weight assignment for the combinational models, such as equal weight, neural-network, genetic programming and etc. In this, an abstract description of how to use AdaBoosting to obtain the dynamic weighted linear combinational model (ACM) of several SRGMs is taken. The combinational linear model is obtained by using,

$$M_{ACM}(t) = \sum W_m M_m(t)$$

Here, (t) is selected M different SRGMs (denoted by m=1…. M) as the candidate models for the ACM is the combinational weight of the selected model. Fitness function (FF) can be defined according to the estimation method of the parameters of these candidate SRGMs. Here, Maximum Likelihood Estimation is used, FF is equation

$$FF = 1/-\log(ML)$$

Where, ML is the maximum likelihood function.

- *Maximum Likelihood Estimation*

Once a model is specified with its parameters, and data have been collected, one is in a position to evaluate its goodness of fit, that is, how well it fits the observed data. Goodness of fit is assessed by finding parameter values of a model that best fits the data, a procedure called parameter estimation. The general methods to estimate the parameters are least squares estimation (LSE) and maximum likelihood estimation (MLE).
Maximum likelihood estimation begins with writing a mathematical expression known as the Likelihood Function of the sample data. For x € R^n the likelihood function of is defined as

$$L(\theta; x) = f(x; \theta)$$

x is regarded as fixed, and is regarded as the variable for L. The log-likelihood function is defined as

$$I(\theta; x) = \log L(\theta, x)$$

The maximum likelihood estimation is the value of

$$L(\theta\ \hat{}(x)) = \max_{\theta \epsilon \theta} L(\theta, x)$$

The likelihood of a set of data is the probability of obtaining that particular set of data, given the chosen probability distribution model. This expression contains the unknown model parameters. The values of these parameters that maximize the sample likelihood are known as the Maximum Likelihood Estimator.

- *Software Reliability Growth Model Selection*
- The proposed algorithm combines advantages of the algorithms, named "The Method of Software Reliability Growth Models Choice Using Assumptions Matrix" (Kharchenko, 2002) and "An Empirical Method for Selecting Software Reliability Growth Models" (Stringfellow, 2002). We found the approach useful of iteratively collecting data and evaluating the models in Empirical Method (Stringfellow, 2002). However, we believe that it would be better to replace some parts of this method to improve the success of the algorithm. First of all, the method selects the initial set of models intuitively. However, intuitive model selection approach may not always include the correct set of models and may cause best models to never be evaluated. Secondly, this method selects the most pessimistic model from the pool of models left, which predicts the maximum number of remaining failures in the software. However, selecting the pessimistic model may increase the testing time unnecessarily. Also, this method considers Failure Count (FC) models, which predict the number of remaining failures. But sometimes, estimation of next time of failure after the release, for example to check whether the software is likely to survive a mission (Schneidewind, 1997), is more important than the number of remaining failures. This information could be obtained from the Times Between Failures (TBF) models. Besides removing that restriction we propose the "local threshold" concept as a novel approach, to be used in deciding on the release time of software.

The method described in (Kharchenko, 2002) is useful if the time to select a reliability model is limited and the reliability requirement is not so high. But to achieve high reliability and find out a model, which represents the test data best, Assumptions Matrix alone is not adequate since assumptions are often violated in practice. Also use of some assumptions may not be practical. For example, one cannot easily assume the number of failures in the system is finite or infinite. Although it has some disadvantages, Assumptions Matrix could be used to select an initial model set by using only those valid assumptions for software under test.

III. PROPOSED ALGORITHM

The proposed algorithm, which has 17 main parts that are described in detail in the following step**:**

- To determine static usable assumptions.
- To determine failure data format: Failure data can be in either Failure Counts (FC) or Time Between Failures (TBF) format.
- To determine local and release threshold: Local threshold will be used before determining release threshold. The idea behind this threshold is not to check the release threshold and continue with testing if last estimated failure time (number of failures for FC data) is close to the one just observed.
- To collect failure data: Record the failure counts per test interval for FC data or times between failures for TBF data.
- Is testing adequate to apply models: To determine if collected data is adequate to apply software reliability models.
- Initial model selection already done?: Check whether the potential model selection done before. If not; then continue with steps 7 and 8 else continue with step 9.

- Determine dynamic usable assumptions: To analyze collected data and select suitable dynamic assumptions.
- Select Potential Models: Making use of the determined static and dynamic assumptions, select reliability growth model/models from the modified Assumptions Matrix.
- Apply Models, Estimate Model Parameters: Estimate model parameters and evaluate models on test data.
- Do models converge: Continue to collect failure data unless at least one model converges. Sometimes due to the nature of data collected, model parameters cannot be obtained and models diverge. If all the selected models diverge then continue testing to collect more data.
- Take next predictions: To calculate next step predictions of the models.
- Rank Models: We applied following procedure suggested to rank models; Apply a goodness of fit test to determine if the model results fit the input data to a specified significance level.
- If a number of models pass the fitness test.
- If only one model provides a good fit to the data then choose the model.
- If no models provide a good fit to data.
- Is FC Data?: if the failure data is of FC format then continue with step 14 else continue with step 15.
- Is Estimate less than Actual (Estimate < Actual).
- To check local threshold.
- To check release threshold.
- To make release decisions.

IV. Experimental Results and Discussions

Modified Assumptions Matrix is used to select initial set of reliability models considering static and dynamic assumptions. While determining the static assumptions, only those valid assumptions for software under test are used. To obtain the appropriate dynamic assumptions, a data analysis tool is used to determine type of data sets, which have either Poisson or Binomial distribution.

Musa Basic model was always ranked third after the Musa-Okumoto and Geometric models. These three models are Poisson Type and Exponential Class models. The main difference is, Musa Basic model assumes finite number of failures whereas other two models assume infinite number of failures. We have recommended to the practitioners not to use the assumption on the number of failures unless there is an obvious reason. And the test result shows that, not using the number of failures assumptions resulted in more accurate predictions.

Figure1 shows plot the cumulative failures of actual data versus total failures, x-axis shows the cumulative time between failures and y-axis shows the total failures.

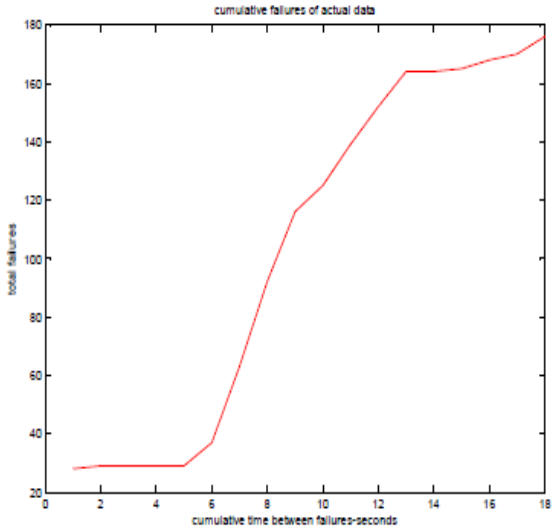

Fig.1. Cumulative failures of actual data

It also included Generalized Poison model, which is a Binomial type model, to test the validity of our assumptions. Yamada S-Shaped, Generalized Poisson and Goel-Okumoto NHPP models were ranked as first, second and third, respectively in the tests. First ranked model predicted the number of remaining failures as just three. If we were selecting best model as the most pessimistic model like the Empirical Method, then we would choose the Goel-Okumoto NHPP model. Because, it has predicted the maximum number of remaining failures.

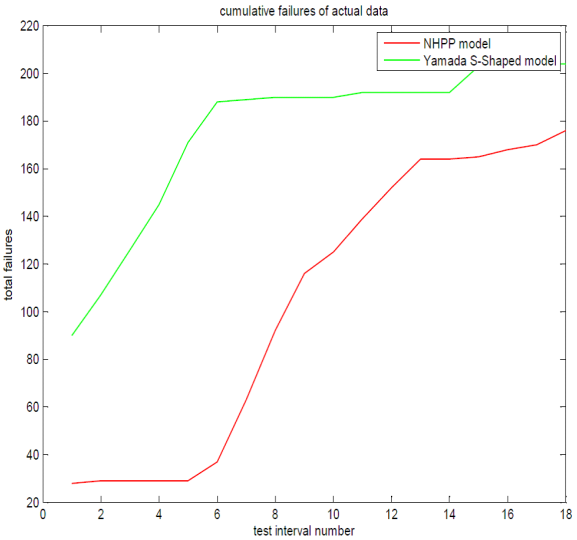

Fig.2. Cumulative failures of actual data, NHPP model and Yamada S-Shaped Model

Figure 2 displays cumulative failures plot of actual data, Yamada S-Shaped model and Goel-Okumoto NHPP model. Cumulative failures plot curve shows that the Yamada S-Shaped model fits to the actual data better than the Goel-Okumoto NHPP model. Also note that total number of failures found in the post-released is slightly more than the predicted number.

CONCLUSION

Software reliability is an important attribute for software quality. It is of prime importance for mission critical systems, which demand high reliability. Thus reliability modeling of software products and predicting reliability via such models at different phases of the software life cycle is of demand. So we have proposed an algorithm for software reliability growth model selection. Proposed algorithm helps to analyze, manage and improve the reliability of the software products. With the help of correctly decided reliability objectives, one can determine when the software is good enough to release and also minimizes risks of deploying the software with serious failures. Moreover, by using the proposed algorithm one can avoid excessive test time and release the product at the correct time with the required reliability.

REFERENCES

1. H. B. Duygulu and O. Tosun, "An algorithm for software reliability growth model selection," Proceeding of IADIS International Conference Informatics 2008.
2. Kharchenko, V.S. et al, 2002, "The Method of Software reliability Growth Models Choice Using Assumptions Matrix," Proceedings of the 26th Annual International Computer Software and Applications Conference.
3. Stringfellow, C., and Amschler, A.A., 2002, "An Empirical Method for Selecting Software Reliability Growth Models." Empirical Software Engineering, 7, 319-343.
4. Haifeng Li, Min Zeng, Minyan Lu, "Exploring AdaBoosting Algorithm for Combining Software Reliability Models". ISSRE 2009.
5. E. X. Cai, M. R. Lyu, "Software Reliability Modeling with Test Coverage Experimentation and Measurement with a Fault-Tolerant Software Project". ISSRE, 2007: 17~26.
6. C. Y. Huang, S. Y. Kuo, M. R. Lyu, "An assessment of testing effort dependent software reliability growth models". IEEE Transactions on Reliability, 2007, 56(2): 198~211.
7. Lyu, M. R, Nikora, A. "Applying Reliability Models More Effective". IEEE Software, 1992, 9(4): 43-52.
8. Y. S. Su, C. Y. Huang, "Neural-network-based approaches for software reliability estimation using dynamic weighted Combinational models" The Journal of Systems and Software, 2007, 80:606-615.
9. C. J. Hsu, C. Y. Huang, "Reliability analysis using weighted combinational models for web-based software". [Online] Available: www 2009, 1131~1132.
10. Eduardo Oliveira Costa, Silvia R. Vergilio, Aurora Pozo, Gustavo Souza, "Modeling software reliability growth with Genetic Programming". ISSRE, 2005: 1~10.
11. Eduardo Oliveira Costa, Gustavo Souza, Aurora Pozo, Silvia R. Vergilio, "Exploring Genetic Programming and Boosting Techniques to Model Software Reliability". IEEE Transactions on Reliability, 2007, 56(3): 422-434.
12. S. H. Kan, Metrics, "Models in Software Quality Engineering". 2nded : Addison-Wesley, 2003.
13. S. Yamada, J. Hishitani, S. Osaki, "Software reliability growth model with Weibull testing effort: a model and application". IEEE Trans. Reliability, vol. R-42, pp. 100– 105, 1993.
14. S. M. Li, Q. Yin, P. Guo, M. R. Lyu, "A hierarchical mixture model for software reliability prediction". Applied Mathematics and Computation, 2007, 185: 1120~1130.
15. In Jae Myung, "Tutorial on maximum likelihood estimation". Journal of Mathematical Psychology 47 (2003) 90–100.
16. Yashwant K. Malaiya, Michael Naixin Li, "Software reliability growth with test coverage". IEEE Transactions on Reliability, 2002, 51(4): 420~426.

AUTHOR'S PROFILE

Tajinder Kaur
Asst. Prof., M.Tech. (IT)
SBBSIET College, Jalandhar
Email ID : kaurtajinder07@gmail.com

Jasjeet Kaur
M.Tech. (CSE)
SBBSIET College, Jalandhar
Email ID : jparmar21nice@gmail.com