Approach to improving the reliability prediction for Software Systems

Jamal Krini*1, Ossmane Krini2, Abderrahim Krini3, Josef Börsök4

Accepted 10th August 2015

DOI: 10.18100/ijamec.24054

Abstract: For example, software is used in the automobile industry, in aerospace engineering and in medical technology. These scopes are considered to be safety-critical. In the case of failure human life as well as its environment might be in danger. Therefore, software reliability plays a huge role in safety critical applications. They provide information on how long and with the help of which resources a software needs to be tested in order to reach a desired reliability. Currently, there are more than 100 reliability growth models existing. The distinction between the models is based on different assumptions. Most of the models are simply improvements of other models. The littlewood model asserts, errors leading earlier to a breakdown possess a higher hazard rate than errors leading much later to a breakdown. This model should be assumed by means of quality-factor improves the prognosis.

Keywords: reliability, littlewood model, software, safety, reliability, growth models, probability of failure, hazard rate, gamma distribution, density function.

1. Introduction

The reliability of a product or a technical system is characterised by its ability to make reliable claims regarding the functions of a product or system at a particular time interval. The term „reliability” is a familiar term in the field of functional safety. Hereby, the term is linked with the proper functioning/functionality of a product. A reliable system is a product, whose functioning is assured at all times and under all conditions. As safety can also be reached when reducing the intended function and assuring a safe condition if necessary, one can also talk about the safety integrity of a system.

For example, software is used in the automobile industry, in aerospace engineering and in medical technology. These scopes are considered to be safety-critical. In the case of failure human life as well as its environment might be in danger. Therefore, software reliability plays a huge role in safety critical applications. It becomes critical when, for example, a software error costs lives. For example, in treating patients suffering from cancer, a machine for radiotherapy had miscalculated the dosage and this has resulted in the loss of eight lives with another 20 people heavily injured. This provides us with another example, demonstrating the danger of software errors. In order to prevent this from happening, software is extensively tested. In this process, various methods exist to make software even more reliable and for this reason, reliability growth models are used.

2. Forecast models

Based on a forecast it is possible to make certain statements about future events. “Prognosis” originates from the Old Greek and means „to predict”. Weighing all relevant and known mistakes, a prognosis tries to predict a future development of a certain phenomenon or a certain situation. Last but not least it means providing an estimation. A very important prediction is the reliability prediction. They provide information on how long and with the help of which resources a software needs to be tested in order to reach a desired reliability. Currently, there are more than 100 reliability growth models existing. The distinction between the models is based on different assumptions. Most of the models are simply improvements of other models. The littlewood model asserts, errors leading earlier to a breakdown possess a higher hazard rate than errors leading much later to a breakdown.

Examples for complexity are:
- Shorter development periods
- Reduced development costs
- Higher level of complexity
- Wider range of functionalities.

Examples for requirements are:
- Minimisation of failure costs
- Increasing product liability
- Increased customer requirements.

Another important factor for functional safety is the probability of failure. It indicates the probability for failure or destruction of a system. The term ‘failure’ stands for cutting out the performance of a task. It results in shifting from workable to corrupt condition. The term ‘fault’ represents the non-performance of at least one of the given requirements.

Reasons for the failure of software can be classified into two causes of failure:
- Operating errors
- Inherent faults
Whereas reasons for the failure of hardware can be classified into three causes of failure:
- Operating errors
- Inherent faults
- Physical faults

Examples for operating errors can be human failures as well as application errors. An inherent fault can be an implementation fault. A physical fault can be caused by attrition, consumption as well as by the aging process. Physical faults can only occur with hardware. Software does not have any attrition or aging process.

Next figure illustrates the failure causes graphically.

### 3. Mathematical basics

Nowadays, more and more complex systems do exist. Mostly they consist of software components. As a logical consequence, the probability of failure of a software component increases. This can lead into catastrophic results.

In order to get a reliable prediction, it is important to resort to the default data. These default data do represent the valid default data in reality, which have been collected during a software test. The following table serves to show the default data that have been detected after a test.

| Failure- number | TBF | Cumulative number of failures |
|-----------------|-----|------------------------------|
| 1               | 45  | 45                           |
| 2               | 53  | 98                           |
| 3               | 19  | 117                          |
| 4               | 62  | 179                          |
| 5               | 34  | 213                          |
| 6               | 112 | 325                          |
| 7               | 268 | 593                          |
| 8               | 102 | 695                          |
| 9               | 251 | 946                          |

MTTF stands for the mean time to failure. MTBF stands for the mean time between two failures. MTTR stands for the mean time to repair. This is equivalent to the time needed for detecting and repairing a failure. The lower figure illustrates the relation between MTTF, MTTR and MTBF.

The cumulative number of default data is represented by the sum of the individual TBFs "Time Between Failure". An important aspect of the reliability prediction is the hazard rate. The hazard rate serves to represent the current intensity level of the failure rate. Furthermore, it is often referred to as default rate or failure rate. The hazard rate is defined as follows:

\[ z(t) = \lim_{\Delta t \to 0} \frac{R(t) - R(t + \Delta t)}{\Delta t} = \frac{1}{R(t)} \int_0^t \frac{dR}{dt} \]

\[ R(t) \] is the reliability of a system. For reliability one has:

\[ R(t) = e^{-\lambda t} \]

Failure density is the temporal derivation of probability of failure

\[ f(t) = \frac{df(t)}{dt} = -\frac{dR(t)}{dt} \]

Thus it follows for the hazard rate:

\[ z(t) = f(t) = \frac{1}{R(t)} = 1 - F(t) \]

Through the derivation of the probability of failure, the failure density function can be determined. For the failure density function is obtained:

\[ f(t) = z(t) \cdot e^{-\int_0^t z(s)ds} \]

For non-repairable systems applies:

\[ \lambda(t) = z(t) \]

For a constant hazard rate applies:

\[ \lambda(t) = z(t) \]

For all other cases applies:

\[ \lambda(t) \neq z(t) \]

The program hazard rate depends is time-dependent and it also depends on the history of the process. The history of the process concentrates merely on the number of errors that have been detected and removed before the point in time \( M(t) \).

The following may be applied for the program hazard rate:

\[ z(t, V_i) = (u_0 - M(t)) \cdot z_0(t) \]

With the help of the mean value function the expected number of detected and removed errors before a point in time \( M(t) \) may be indicated.

\[ \mu(t) = E[M(t)] \]

The following applies with regards to that:

\[ E[M(t)] = u_0 \cdot F_a(t) \]

Therefore, the following applies for the mean value function:

\[ \mu(t) = u_0 \cdot F_a(t) = u_0 \cdot \left[ 1 - e^{-\int_0^t z_a(s)ds} \right] \]

The intensity of failure corresponds to the expected value of the program hazard rate. At the same time it corresponds to the mean value function after the point in time.

\[ \lambda(t) = E[z(t, V_i)] = \frac{d\mu(t)}{dt} \]

### 4. Reliability models

The Jelinski- Moranda- Model is one of the first models and was published in 1972. For this model, Jelinski and Morand made the following assumptions:

- the amount \( A_0 \) of unknown mistakes in a software is firm
- every fault is just as dangerous given the probability of immediately making a mistake
- the hazard rate of a mistake does not change over time and stays constant
- the times in between mistakes stays
- default is corrected immediately

As can be seen, in this model a constant hazard rate \( z_a(t) = \phi \) is
assumed for any errors, which is unrealistic. A better model is at this point the Littlewood model. The model asserts, errors leading earlier to a breakdown possess a higher hazard rate than errors leading much later to a breakdown. Thus, the hazard rate must decrease with time, regardless of whether an error was detected or not. Consequently, the program hazard rate does not drop steadily during breakdowns like with the Jelinski-Moranda-Model, but it rather tends to gradually decrease. This effect may be presented with the help of a Binominal Model. The littlewood model considered exactly this phenomenon.

In reliability engineering, the gamma distribution helps describe the time until the “i-th” failure, assuming that times are exponentially distributed. For the gamma distribution, the probability density function (PDF) may be defined as the following:

\[ f(x) = \begin{cases} \frac{\lambda^i}{\Gamma(i)} x^{i-1} e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases} \]

Thereby, \( \lambda \) corresponds to the failure rate and \( \Gamma(i) \) corresponds to the gamma function. The gamma function is defined as the following:

\[ \Gamma(i) = \int_0^\infty x^{i-1} e^{-x} dx \]

For \( i = 1 \), the gamma distribution leads to the exponential function. The gamma distribution \((\alpha, \beta)\) is used in order to assign a higher hazard rate to errors occurring earlier and in order to assign a lower hazard rate to errors occurring later, thus preventing that error rates do not remain constant but that they rather depend on the use of the program. In this process, random draws from the gamma distribution \((\alpha, \beta)\) are used for the hazard rates. If the program is run for a total time period \( t \), the following time-dependent hazard rate will be applied by means of the Bayes’ theorem:

\[ z_a(t) = \frac{\alpha}{\beta + t} \]

By means of this hazard rate, requirements from the start are met. While time increases, the hazard rate sinks. As a result, it can be said that the higher the hazard rate, the earlier an error occurs. In case at least one failure has occurred until the point of time \( t \), the distribution function can be calculated as the following:

\[ F_a(t) = 1 - e^{\frac{-\alpha}{\beta + t}} \]

After the integral equation has been solved, it becomes evident that a Pareto distribution is followed by a distribution function:

\[ F_a(t) = 1 - \left( \frac{\beta}{\beta + t} \right) \alpha \]

The following applies for the program hazard rate:

\[ z(t, m(t)) = (u_0 - m(t)) \cdot z_a(t) = (u_0 - m(t)) \cdot \frac{\alpha}{\beta + t} \]

The mean value function is calculated as follows:

\[ \mu(t) = u_0 \cdot \int_0^t F_a(s) ds = u_0 \cdot \left( 1 - e^{\frac{-\alpha}{\beta + t}} \right) \]

By considering the Limes of the mean value function for \( t \rightarrow \infty \), proof for this model being an infinite model is provided. Furthermore, the following applies for the failure intensity:

\[ \lambda(t) = \frac{d\mu(t)}{dt} = \frac{d}{dt} \left( u_0 \cdot \left( 1 - \left( \frac{\beta}{\beta + t} \right) \right) \right) = u_0 \cdot \frac{\alpha}{\beta + t} \left( \frac{\beta}{\beta + t} \right) \]

Calculating the Limes and the failure intensity, the following applies:

\[ \lim_{t \rightarrow \infty} \lambda(t) = 0 \]

Next Figure shows the program hazard rate \( z(t, m(t)) \).

An important part of reliability prediction is the estimation procedure. It is used to determine the model parameters. The Maximum Likelihood Estimation (MLE) is the most effective and most common method in use. It is defined as follows:

\[ L = L(\phi) = \prod_{i=1}^n P(X_i = x_i|\phi) \]

For the failure density function follows:

\[ f(t_i) = z(t_i)e^{-e^{-\frac{\alpha}{\beta + t}}t_i ds} = (U_0 - i + 1) \frac{\alpha}{\beta + t} \]

For the parameter estimation of the proposed Littlewood model follows:

\[ L = L(U_0, \alpha, \beta) = \prod_{i=1}^n f(t_i) \]

By taking the logarithm follows:

\[ \log \left( L(U_0, \alpha, \beta) \right) = \sum_{i=1}^n \log(U_0 - i + 1) \]

\[ + n \cdot \log(\alpha) \]

\[ - n \cdot \log(\beta + t) \]

\[ - \sum_{i=1}^n a \cdot \log(U_0 - i + 1) \cdot \log(\beta + t) \]

\[ + \sum_{i=1}^n a \cdot \log(U_0 - i + 1) \cdot \log(\beta) \]

Now the parameters can be estimated.

5. Improve predictive Model with Quality- Factor

In order to improve the prediction accuracy the “Hold-On” method has to be applied. Firstly, the software needs to be tested. A table including the default data will be created after the test. These default data will be taken and they are going to be attached to the Littlewood model.
The data procession block serves for the data measurement value. It is preferable that not all data that is available is evaluated at the same time. Moreover, the goal is to piecewise analyze the data. After the first data amount has been analyzed, and a prediction of reliability is has been made, it can be compared to a real second data amount. By comparing it, the quality factor (Q-factor) may be determined.

Using this quality-factor an improved prognosis should be created. One potential method is to multiply the hazard rate with the quality factor γ:

For the hazard rate follows:

\[ z_a(t) = \frac{\alpha}{\beta + t} \cdot \gamma \]

For the distribution function follows:

\[ F_a(t) = 1 - \left( \frac{\beta}{\beta + t} \right)^{\alpha \cdot \gamma} \]

For the program hazard rate follows:

\[ z(t, m(t)) = (u_0 - m(t)) \cdot z_a(t) = (u_0 - m(t)) \cdot \frac{\alpha \cdot \gamma}{\beta + t} \]

The mean value function is therefore calculated as follows:

\[ \mu(t) = u_0 \cdot F_a(t) = u_0 \left( 1 - e^{-\int_0^t z_a(s)ds} \right) = u_0 \left( 1 - \left( \frac{\beta}{\beta + t} \right)^{\alpha \cdot \gamma} \right) \]

Subsequently, it can be demonstrated by means of comparison whether the required quality factor would lead to an improvement of the prediction.

6. Summary

Currently, there are more than 100 reliability growth models existing. The distinction between the models is based on different assumptions. Most of the models are simply improvements of other models. The littlewood model asserts, errors leading earlier to a breakdown possess a higher hazard rate than errors leading much later to a breakdown. This model should be assumed by means of quality-factor improves the prognosis.

References

[1] J. Börcsök, 2004. Electronic Safety Systems, Hardware Concepts, Models, and Calculation. Heidelberg, Germany 2004.
[2] J.D. Musa, Software Reliability Engineering 2004. More Reliable Software Faster and Cheaper. United States of America 2004.
[3] N. F. Schneidewind, Software Reliability Engineering 1992.
[4] J. Börcsök, O. Krini, 2009. Principle Software Reliability Analysis with different Failure Rate Models. IEEE Int. Symp. ICAT 2009
[5] B. Lindqvist, A. Doksum, 2003. Mathematical and Statistical Method in Reliability. Covent Garden, London 2003.
[6] M. Takahashi and Y. Kamayachi, IEEE Computer Society, 1991. An Empirical Study of a Model for Program Error Prediction, in Software Reliability Models. pp. 71-77.
[7] Musa, J.D.: “Software Reliability Engineering”, McGraw-Hill, New York, 1998.
[8] A.L. Goel and K. Okumoto, “Time Dependent Error Detection Rate Model for Software Reliability and Other Performance Measures”, IEEE Trans. Rel., R-28(3), pp. 206-211.
[9] J.D. Musa, A. Iannino and K. Okumoto, Software Reliability: Measurement, Prediction, Application, McGraw-Hill, 1987.
[10] O.Krini, J.Börcsök: “Different Approaches to Predict Software Reliability”, Proceedings of 24th international congress, 2011
[11] Lyu, M. (ed.): "Handbook of Software Reliability", McGraw-Hill, New York, 1996.
[12] Huan-Jyh Shyr: “A stochastic software reliability model with imperfect-debugging and change-point”, The Journal of systems and software, 2003.
[13] IEEE Recommended Practice on Software Reliability, Standards Committee of the IEEE Reliability Society, 2008.
[14] X.Zhang, X.Teng, H.Pharm: “Considering Fault Removal Efficiency in Software Reliability Assessment”, IEEE vol.33, 2003.
[15] N.F. Schneidewind, “Analysis of mor processes in computer software,” Sigplan Notices, Vol. 10, pp.337-346, 1975.
[16] Lyu, M. (ed.): "Handbook of Software Reliability", McGraw-Hill, New York, 1996.
[17] A. Birolini, “Reliability Engineering: Theory and Practice”, Springer Verlag, Heidelberg, 2007.

Minimizing risk in Applying Metrics on Multiple Projects, Proc. IEEE Int. Symp. pp. 173-179.

This journal is © Advanced Technology & Science 2013