Optimisation of software reliability prediction

JAMAL KRINI, OSSMANE KRINI AND JOSEF BÖRCSÖK
Department of Computer Architecture and System Programming
University of Kassel
Wilhelmshöher Allee 71 – 73, 34121 Kassel
GERMANY
j.krini@uni-kassel.de  ossmanekrini@yahoo.de http://www.uni-kassel.de/eecs/rs

Abstract: - The reliability of software is one of the most important software metrics. In the course of time, systems have become more and more complex and that is why software has become more complex, too. It is an undeniable fact that this will lead to an increasing number of software faults. Models of software reliability are used to track down software faults. The aim is to make sure that an improvement in reliability prediction is gained. This paper explains the reasons for software and hardware failures. Furthermore, it describes a procedure to establish an innovative and improved model of prediction for software. The predictions will be based on realistic model assumptions. An appropriate model for the particular project can be determined by the Q factor. In this paper, different possibilities of evaluating a quality criterion are used.

Key-Words: - reliability, software, hardware, faults, prediction, failure, Q factor, reliability growth models, prognosis, probability, safety, MTTF (mean time to failure), MTBF (mean time between two failures), MTTR (mean time to repair), hazard rate.

1 Introduction

At the National Cancer Institute in Panama City, a software fault has led to the death of eight patients after radiotherapy, and the injury of twenty others caused by overdose. The doctors in charge had to stand trial.

Ariane 5 is a European launch vehicle of the Ariane series that has been developed by order of the ESA and has been in use since 1996. The maiden voyage of Ariane 5 took place on 4th July, 1996. The launch of this rocket ended in a disaster. It exploded after exactly 36.7 seconds. The extent of damage amounted to one billion Euro. The rocket had been developed in a period of 10 years at a cost of six billion Euros. Immediately after this tragedy, an investigative commission was convened who found out that it had been caused by a software fault. The explosion had been caused by a corrupt conversion from a 64-bit to a 16-bit number.

These incidents show that corrupt software can cost both money and human lives. Reliability growth models are used to reduce such software faults.

2 Mathematical and statistical principals of safety theory

Nowadays, technical systems in areas of critical safety contain not only mechanical and electrical hardware, but increasingly also microcontrollers and software. Therefore, the aspect of safety is very important. The word ‘safety’ is one of the most important terms in safety technology. Reliability growth models are used to identify and prove reliability requirements. Reliability is the probability of a function's performance in a specific time interval and under specific working conditions. Whereas safety is the ability not to cause and to prevent danger with given limits and given periods.

Influencing factors on reliability are both complexity and the requirements.

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.
The condition of a technical system can be either workable or not workable:

\[ x(t) = \begin{cases} 1, & \text{if system workable at time } t \\ 0, & \text{if not} \end{cases} \]  
(1)

The operational lifespan of a technical system is the interval \( T \) between the initial start-up \( t=0 \) and the failure \( t=T \). Therefore for the condition of a system one has:

\[ x(t) = \begin{cases} 1 & \text{f ü r } t < T \\ 0 & \text{f ü r } t \geq T \end{cases} \]  
(2)

Reliability is an important factor for functional safety. As mentioned before, reliability is a criterion for the probability of a function’s performance in a specific time interval. For reliability one has:

\[ R(t) = e^{-\lambda t} \]  
(3)

Another important factor for functional safety is the probability of failure. It indicates the probability for failure or destruction of a system. For the probability of failure one has:

\[ F(t) = 1 - e^{-\lambda t} \]  
(4)

Failure density is the temporal derivation of probability of failure. This is given in equation 5. The sum of reliability and probability of failure is always 1 (see equation 6). In figure 1 you can see the related graphic.

\[ f(t) = \frac{dF(t)}{dt} \]  
(5)

\[ F(t) + R(t) = 1 \]  
(6)

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. Figure 2 illustrates the relation between MTTF, MTTR and MTBF.

\[ MTTF = \int_{0}^{\infty} t \cdot f(t) \, dt = \int_{0}^{\infty} R(t) \, dt \]  
(7)

Figure 2: Correlation between MTTF, MTTR and MTBF

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.

Figure 3 illustrates the failure causes graphically.
3 Reliability Growth Models

The term ‘prognosis’ derives from the Old Greek and signifies as much as ‘predicting’. A prognosis tries to foresee the future development of a specific phenomenon or a specific situation by assessing all relevant and established factors. So it is merely an evaluation. Reliability prediction is a very important type of prediction. Reliability growth models are used to predict a reliability prediction for the software. In recent years a great number of reliability growth models were published. The models differ in the impelling resource that is applied, in the type of their influence and in the distribution of durability. Most reliability growth models are merely expansions of other models.

The models can be classified by five attributes:

- Time domain
  - Calendar time
  - Execution time
- Category
  - Limited number of failures
  - Infinite number of failures
- Type
  - Binomial distribution
  - Poisson distribution
- Class (only with regard to a limited number of failures)
  - Exponential
  - Gamma
- Family (only with regard to an infinite number of failures)
  - Inverse linear
  - Geometrical

Models with a limited number of failures include, for example, the Musa basic execution time Model and the Jelinski Moranda Model. Models with an infinite number of failures include, for example, the Musa Okomoto Model. The Model can be divided into time-dependent and Time-independent:

- Time-dependent models (Data/test results refer to time):
  - Shooman-Modell
  - Jelinski-Moranda-Modell
  - Poisson-Modell
  - Schick-Wolverton-Modell
- Time-independent models (Data/test results do not refer to time)
  - Mill-Modell
  - Lipow -Modell
  - Rudner-Modell
  - Nelson-Modell

Selecting one of these models is project specific. Therefore at no time a suitable model can be selected beforehand.

It is necessary to make assumptions about the systems that are looked at and their behavior as well as about the process of creating it, in order be able to quantify the reliability of software systems. These assumptions lead to new models predicting software reliability being developed. These models may be determined by procedures of the probability theory. One of the first of such models was the Jelinski-Moranda-Modell.

For this model, Jelinski und Moranda developed these hypothesis:

- The amount $u_0$ of the unknown errors in the software is set.
- every fault is equally dangerous in terms of the probability of causing another error
- the hazard rate of every error does not change over time and remains constant
- the times between errors are not dependent
- an error is immediately corrected

However, the number of errors in the operating period and the duration of operation between two errors that follow each other needs to be available as data. The model assumes that the rate of finding a
mistake is proportional to the number of residual errors. This number continually decreases concerning this model. The Generalized-Poisson Modell is similar to the Jelinski- Moranda Modell. However, there are differences in the method of counting errors. The Schneidewind Modell as well is very similar to the Jelinski-Moranda Modell. This model assumes that the process of finding mistakes changes to a progress of the test and that newly counted mistakes are of higher weight.

An important part of reliability prediction is the estimation procedure. It is used to determine the model parameters. The most established methods are:

- Method of moments
- Least squares method
- Maximum Likelihood Estimation (MLE)

The Maximum Likelihood Estimation (MLE) is the most effective and most common method in use. It is defined as follows:

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

(8)

To estimate the model parameters according to MLE, the following steps are performed:

- Set the likelihood function
- Form the natural logarithm of the likelihood function
- Differentiate the likelihood function with respect to every guessable parameter

4 Research approach

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.

The aim of the approach is to guarantee an improvement of the reliability prediction on the basis of reliability models that already exist. Thus, repair and maintenance costs can be possibly saved, which is very economic. Moreover, a more precise number of system failures can be estimated with an improved reliability prediction. Figure 4 illustrates roughly illustrates a basic concept of the research work.

In this research work, with the help of distribution functions, three innovative software prediction models shall be set up that should be able to provide a better prediction than former customary software reliability models. This can be achieved by choosing a more suitable distribution function to be able to describe the hazard rate in a better way. This improvement results in a better prediction. Furthermore, the hazard rate of the time of fault occurrence can be taken into account. A fault that occurs in the infinite is of little importance for the software prediction model, because it cannot occur. Consequently the hazard rate decreases although no
fault has occurred or has been detected.

The best model is to be chosen according to experience and field of application. However, none of the models can predict well for every single data set. Furthermore, it is not beyond the bounds of possibility to be able to decide with confidence ahead of the assessment which model fits the prediction best. In addition, a method shall be used that can decide, by means of the Q factor, which one of the various reliability models is considered applicable. This can be enabled by comparing the Q factors.

First of all, a prediction is made by the use of the first failure data (first data set) of the entire failure data packet (see figure 4). Here, the instant of the following failure is estimated. This will be estimated with the three innovative software prediction models. In this way, the estimated instant of failure of the following failure can be compared with the actual instant of failure of the following failure. Thus it is possible to calculate the Q factor. This procedure is gradually repeated by adding to the data out of the entire failure data packet set one instant of failure after the other. Various assessment possibilities shall be used for the quality criterion.

- Absolute relative forecast errors
- Measurements of variability
- Prequential likelihood function
- δ plot and ξ plot

Systematic faults can be corrected by means of measuring the Q factor to acquire an improvement of the prediction. Recalibration can be explained like this: A gun that pulls to the right with every shot can be corrected by a left shift. Consequently, the measures of recalibration will be used for error correction.

Because you cannot decide with certainty ahead of the assessment which model fits the prediction best, the three innovative software prediction models shall be used for each one of the assessment possibilities mentioned above.

5 Conclusion

Software reliability models are part of the most important components of the functional safety. Therefore, it is absolutely necessary to carry on research in this field in order to keep creating more and better forecast models. In this paper, a new process model is presented that shall provide a better predictive ability. For this purpose three innovative software prediction models are compiled, which shall provide a better prediction than former customary software prediction models. Moreover, the available failure data are divided. For every data set an estimation shall be made by the use of the three innovative software prediction models. The estimated figure is compared to the actual, available figure. This is a good way to detect the most suitable of the three innovative software prediction models.

6 Reference


