A New Method to Detect and Correct the Critical Errors and Determine the Software-Reliability in Critical Software-System

Ossmane Krini and Josef Börcsök
Department of Computer Architecture and System Programming University of Kassel, Germany
E-mail: o.krini@uni-kassel.de, j.boercsoek@uni-kassel.de

Abstract. In order to use electronic systems comprising of software and hardware components in safety related and high safety related applications, it is necessary to meet the Marginal risk numbers required by standards and legislative provisions. Existing processes and mathematical models are used to verify the risk numbers. On the hardware side, various accepted mathematical models, processes, and methods exist to provide the required proof. To this day, however, there are no closed models or mathematical procedures known that allow for a dependable prediction of software reliability. This work presents a method that makes a prognosis on the residual critical error number in software. Conventional models lack this ability and right now, there are no methods that forecast critical errors. The new method will show that an estimate of the residual error number of critical errors in software systems is possible by using a combination of prediction models, a ratio of critical errors, and the total error number. Subsequently, the critical expected value-function at any point in time can be derived from the new solution method, provided the detection rate has been calculated using an appropriate estimation method. Also, the presented method makes it possible to make an estimate on the critical failure rate. The approach is modelled on a real process and therefore describes two essential processes - detection and correction process.

1. Introduction
The two processes above are described as two differential equations. The new approach performs a detection of critical errors as well as a prediction after a repair time. If a critical error is detected, a certain amount of time is required to fix it. The time difference also is introduced as a mathematical time-dependent function.

This approach provides following advantages and differences to existing methods:
- One aspect is the newly introduced approach which uses one of the conventional reliability models to estimate the total error count and requires real failure data to ultimately make a prediction regarding the critical errors
- Instead of making a prognosis on critical errors by using already known models, which is often not possible, it suffices to base a better prognosis on a relatively low number of errors using the new approach.
- The currently used software reliability models frequently consist of very unrealistic model assumptions. These model assumptions can neither be used for the critical error prediction or
for the error analysis. In the new approach, the repair time will be explicitly described and applied.

- With the repair time, the correction process is described and adjusted to the new method. By doing this, a realistic prediction of critical errors can be conducted.

Increased functionality is identified in current and future software-systems. Statistics and analysis of failures of such complex software-systems reveal that more than half of the system failures are attributed to the software components. Therefore, it is necessary from a scientific and economic standpoint to use reliability models that are capable of predicting software failures. Software reliability is the probability of a failure-free function of a computer program at a specific time in a specific environment. Hardware components are assumed to only fail spontaneously. The failures are due to manufacturing defects, wear, or external factors. Errors in software that cause failures exist statically and are permanent. There are no random malfunctions in software like there are in hardware. The random occurrence of incorrect software behavior usually occurs through the many variations and combinations of software. Today, many programs exist to evaluate and predict hardware reliability. However, even to this day, there are no reliable and meaningful software probability models available to make a general prognosis for safety-critical faults. Since there is no preferable or particular software reliability model, software developers are forced to test a number of individual models. The predictions resulting thereof, however, do not always offer the desired level of accuracy.

2. Conventional Software Reliability Models for Prediction of Software-errors

Intuitively, a software-system is understood to be reliable when the reasonable user’s expectations are met without failures throughout a time period. This time period is represented with the \( T \) variable. Mathematical software reliability models aim to quantify the reliability. Such models are well founded in theory and are used particularly for safety critical systems. This means that the random variables \( T_1, T_2...T_n \) for the time between the failures can still be seen as exponentially distributed, but the parameters of the distribution \( \lambda_1, \lambda_2...\lambda_n \) can be different. Software reliability models (SRM) make certain assumptions as to how the parameters interact. About eighty such models are known in literature [1]. The purpose of such software reliability models is the prognosis of the expected reliability of a system based on data from the past. The future failure rate can be and is predicted from the known times between failures and until a point in time or \( T \). It is assumed that the pure operating time is measured; this means that the accumulation of operating time is stopped as soon as the system is out of operation or in test mode.

2.1. Binomially distributed error models

The first binomially distributed models were developed by Jelinski and Moranda and were based on birth and death processes. A binomial distributed model has an unknown set number of errors \( u_0 \). Every error has the same probability of occurrence. The hazard rate of every error \( z_a(t) \) remains the same over time and is constant (\( \Phi \)). Furthermore, a binomially distributed error model assumes that the time between failures are independent from each other. If a failure occurs at point in time \( t_s \), it is corrected immediately at point in time \( t_s \). In this approach, it is assumed that an error correction does not lead to new errors. This is why Jelinski and Moranda introduced the following expression for the failure rate:

\[
\dot{\lambda}(t) = c \cdot N_i, \quad t \in [t_{i-1}, t_i]
\]  

(1)

where \( c \) is a constant and \( N_i \) is the residual error number at point in time \( t_i \). Thus, following can be written with the Equation (1):
The failure rate of a binomially distributed error model can be defined as step function [1].

\[
\dot{\lambda}(t_i) = c \cdot N_1 = c \cdot N
\]

\[
\dot{\lambda}(t_2) = c \cdot N_2 = c \cdot (N_1 - 1) = c \cdot (N - 2 + 1)
\]

\[
\dot{\lambda}(t_3) = c \cdot N_3 = c \cdot (N_2 - 1) = c \cdot (N - 3 + 1)
\]

.......... The failure rate of a binomially distributed error model can be defined as step function [1].

\[
\dot{\lambda}(t) = c \cdot (N - i + 1)
\]

Software reliability \( \dot{R}(t) \) is defined according to Musa is as follows:

\[
\dot{R}(t) = e^{-\int_0^t \dot{\lambda}(\tau) d\tau} = e^{-\int_0^t c(N-i+1)d\tau} = e^{-c(N-i+1)t}
\]

The software failure probability \( \dot{F}(t) \) thus results for:

\[
\dot{F}(t) = 1 - e^{-\int_0^t \dot{\lambda}(\tau) d\tau} = 1 - e^{-\int_0^t c(N-i+1)d\tau} = 1 - e^{-c(N-i+1)t}
\]

To make an estimate for the unknown parameters \( N \) and \( c \), the maximum-likelihood-method is used. This method determines the parameters so that the mean value function, also called the expectation value, is maximized.

2.2. Poisson distributed Error Models

The Poisson distributed Error Model is the best-known model among software reliability models. This model has been developed by Musa at the AT&T Bell Laboratories. Musa was an important advocate for the use of models and thus, his models are used in various areas. The strengths of this model are in the estimation of test effort in relation to the execution time to achieve a specified MTBF (mean time between failure). \( M(t) \) is the number of failures in the time interval from \([0...t]\). \( M(t) \) is a Poisson process with mean value function:

\[
\dot{\mu}(t) = \beta_0 \cdot \left( 1 - e^{-\beta_1 \cdot t} \right)
\]

where \( \beta_{0,1} > 1 \). The expectation value is a cumulative function. Therefore, the function increases continuously. Since \( \lim_{t \to \infty} \dot{\mu}(t) = \beta_0 \) applies, the Musa-execution-model belongs to the finite software reliability models. The parameter \( \beta_0 \) represents the total error count. The execution times between the failures are piecewise exponentially distributed and the errors that cause failures are independent from each other. Furthermore, Musa assumes that an error is corrected immediately, without causing a new software error. The mean function is defined as [2]:

\[
\dot{\mu}(t) = E(M(t)) = \sum_{n=0}^{\infty} n \cdot P(M(t) = n).
\]

This results to the software failure rate:

\[
\dot{\mu}'(t) = \frac{d}{dt} E(M(t)) = \beta_0 \cdot \beta_1 \cdot e^{(-\beta_1 \cdot t)}
\]
and must decrease monotonously. \( \mu'(t) < 0 \), for instance, is sufficient. It is also possible that \( \mu'(t) \) deviates. However in this case, at least the envelope curve of the peak values must fall monotonously. When the parameter \( \beta \) becomes too large, the failure rate decreases faster and runs to zero. If, however, the parameter becomes too small, the failure rate decreases slower. In all cases, though, the failure rate function goes towards zero. In practice, the Muse execution model, the MTTF-value (Mean-Time-To Failure) value, is not determined, but the mean value function \( \mu(t) \) provides the number of errors during a warranty period. Musa makes the following assumptions:

- The failure rate \( \lambda(t) \) is piecewise constant. The result is that the failure rate between two errors is distributed exponentially.
- \( \lambda(t) \) is not proportional to the residual error count and
- there possibly is a non-linear relationship between time scale and the "natural" time

3. Description of Detection and Correction Process for Prediction of Critical Errors in Software-Systems

Most conventional methods distinguish between error-detection and error-correction. However, to perform a realistic prognosis, the model assumptions are to be modified and adapted. This chapter is going to present a new approach [3]. The detection and correction process for critical errors is mathematically described and fed into a finite as well as an infinite stochastic model. Thus a realistic prediction for the error detection and error elimination will be guaranteed.

3.1. Challenge

One of the known model assumptions is that for every error \( F_a \) that occurs at a particular point in time \( t_a \), is immediately corrected at time \( t_a \). Therefore, there is no repair time \( \Delta t \). Nonetheless, in reality, an error is not corrected immediately after it has been detected. The time that is required to remove a critical error of course depends on other factors including complexity of the detected error. Likewise, the time depends on how quickly the team can reproduce the critical error, find the cause and fix it. Therefore, the repair time during the correction process should not be neglected, figure 3.1.

![Figure 3.1. Constant Delay-time.](image-url)

Most conventional models do not consider the repair time. Therefore, these models make inaccurate forecasts, especially regarding critical errors. Hence, the repair time \( \Delta t \) must be considered in the prediction process [3].
In the first part of this chapter, repair time $\Delta \tau$ is mathematically described and assumed to be constant over time. The constant delay time serves to emphasize the difference to conventional methods.

In the second part, however, the repair time $\Delta \tau$ is considered realistically as a time-dependent function $\Delta \tau(t)$. Subsequently, the delay time is fed into different models in order to make a prognosis regarding the residual number of critical errors.

The maximum-likelihood-method is considered for parameter estimation [4].

### 3.2. Approach for calculating the critical failure probability depending to a delay time $\Delta \tau$

In order to describe a detection and correction process, different requirements apply. Both a detection and a correction process run in a non-homogenous Poisson process. The critical errors are independent from each other. The mean number of critical errors in the time interval $[t, t+\Delta \tau]$ is proportional to the mean number of the remaining not corrected critical errors. Every critical error that occurs is therefore corrected with no new errors added.

The delay time $\Delta \tau$, until the correction process can be assumed, is to be constant. Therefore, the correction process is considered to be a detection process delayed by $\Delta \tau$. The following therefore applies:

$$\hat{\mu}_{c,i}^{(t)}(t) = \hat{\mu}_{c,i}^{(t-\Delta \tau)}$$  \hspace{1cm} (9)

where $\hat{\mu}_c(t)$ is the cumulative function of a critical error and $\hat{\mu}_{nc}(t)$ is the cumulative function of a non-critical error. Figure 3.3 left shows the progression of both functions. It can be seen that the correction curve in blue is shifted by a constant factor $\Delta \tau$.

At point in time $\tau_x$ a certain number of errors $N$ is detected, but corrected at point in time $\tau_y$. Since every critical error requires a different amount of time to be corrected, it is very unrealistic that a delay time $\Delta \tau$ can be assumed to be constant [2]. Though, for a new approach, a non-constant delay time $\Delta \tau(t)$ is chosen, figure 3.3 right.

![Figure 3.2. New algorithmic process.](image-url)
There is no constant delay time $\Delta \tau(t)$ between the detection and correction processes. Now it remains to be clarified how the delay time $\Delta \tau(t)$ is described mathematically. The delay time $\Delta \tau(t)$ can be described at will; however, the delay time depends on the software-system that is going to be analyzed. Hence, the time shift $\Delta \tau(t)$, for instance, can be assumed to be linear:

$$\Delta \tau(t) = t$$

(10)

This would mean, however, that proportionality exists between the abscissa, the time axis, and the ordinate, which is the number of errors. Because of that, a linear delay time $\Delta \tau(t)$ is not recommended for this approach.

Another consideration, which arose during the process of this thesis, is to consider the delay time as an exponential function with a negative exponent:

$$\Delta \tau(t) = a \cdot e^{-t}$$

(11)

The goal is to obtain a cumulative function as a result for the correction process that reflects a relative approximation of reality. Since the “In-func” can only adopt positive values, another promising solution seems to be the selection of a logarithmic function. This could lead to a so-called “learning process”. To ultimately examine the effect of the exponential or logarithmic delay time in the correction process, the detection and correction process must be described mathematically.

The following first order differential equation applies for the critical detection process:

$$\frac{\partial D\mu_c(t)}{\partial t} = d_\psi(t) \cdot \left( \sum_{i=0}^{n} u_{0_i} - s D\mu_c(t) \right)$$

(12)

Function $d_\psi(t)$ describes the detection rate for the critical errors and shall be assumed to be constant in this work. Index “$\psi$” indicates that the detection rate refers to the new approach and the newly introduced constant $\Psi$. $\Psi$ represents the ratio between the critical error and the total number of errors:

$$\Psi = \frac{\sum_{i=0}^{n} u_{0_i}}{\sum_{i=0}^{n} u_{0_i} + \sum_{i=0}^{n} u_{0_{\neg i}}} \begin{cases} u_{0_i} \Rightarrow \text{critical number of faults} \\ u_{0_{\neg i}} \Rightarrow \text{no critical number of faults} \end{cases}$$

(13)
The rate $d_{\psi}(t)$ is to be always seen as unknown and therefore needs to be estimated by using the maximum-likelihood-method. The detection process runs a non-homogeneous Poisson form. Furthermore, all critical errors shall be independent from each other. \( \sum_{i=0}^{n} u_{0,i} \) represents the total number of critical errors. \( \hat{s}_{D\mu_{c}}(t) \) describes the expectation value of the critical errors in the detection process. According to the "postulate" at the beginning of the prognosis \( t = 0 \), there are no critical errors. Because of this, the side condition follows the differential condition [2]:

\[
\hat{s}_{D\mu_{c}}(0) = 0
\]  

(14)

Thus, the inhomogeneous differential equation of the first order can be solved completely.

\[
\frac{\partial \hat{s}_{D\mu_{c}}(t)}{\partial t} = d_{\psi} \sum_{i=0}^{n} u_{0,i} - d_{\psi} \hat{s}_{D\mu_{c}}(t)
\]

(15)

The delay time \( \Delta \tau(t) \) is irrelevant, because, although the critical error has been detected, it has yet to be corrected. In the most cases, the error cannot be corrected immediately because the system is in operation.

Now the correcting process of the critical error is described mathematically as follows:

\[
\frac{\partial \hat{c}_{\mu_{c}}(t)}{\partial t} = c_{\psi}(t) \cdot (\hat{s}_{D\mu_{c}}(t) - \hat{c}_{\mu_{c}}(t))
\]

(16)

\( \hat{c}_{\mu_{c}}(t) \) describes the expectation value of the critical errors. The correction rate \( c_{\psi}(t) \) can also be interpreted using Equation (16):

\[
c_{\psi}(t) = \frac{\frac{\partial \hat{s}_{D\mu_{c}}(t)}{\partial t}}{\hat{s}_{D\mu_{c}}(t) - \hat{c}_{\mu_{c}}(t)} = \frac{\hat{s}_{D\mu_{c}}(t) - \hat{c}_{\mu_{c}}(t)}{\frac{\partial \hat{s}_{D\mu_{c}}(t)}{\partial t}}
\]

(17)

Equation (17) shows, that the correction rate \( c_{\psi}(t) \) can be seen as the error correction rate per detected errors, but as not corrected "critical" errors. In reality, the correction rate depends on the complexity of the problem being analyzed, the abilities of the test team, and the time restrictions for the handover of the finished software to the customer.

Equation (17) can now be solved with the following side condition:

\[
\hat{s}_{c\mu_{c}}(0) = 0
\]

(18)

The solution of the differential equation for the correction process then leads to the following expectation value:

\[
\hat{s}_{c\mu_{c}}(t) = e^{-C(t)} \cdot \left( \int_{0}^{t} \left( \sum_{i=0}^{n} u_{0,i} + \sum_{i=0}^{n} u_{0,\psi_{i}} \right) e^{C(t)} \cdot e^{D(t)} \cdot \left[ 1 - e^{-D(t)} \right] ds \right)
\]

(19)

where:
\[ D(t) = \int_{0}^{t} d(s) \, ds \]
\[ C(t) = \int_{0}^{t} c(s) \, ds \]  

(20)

In order to maintain the mathematical overview and for the sake of simplicity, the correction rate \( \psi (t) \) shall also be assumed to be time-independent in the scope of this work. When the differential equation is now solved, the following expectation value is obtained for the correction process:

\[ \frac{d}{dt} \mu_c(t) = \left( \sum_{i=0}^{n} u_{0_i} \right) \left( 1 - \left( 1 + c \psi \cdot t \right) e^{-c \psi \cdot t} \right) \]
\[ \frac{d}{dt} \mu_c(t) = \left( \sum_{i=0}^{n} u_{0_i} \right) \left( -c \psi \cdot t e^{-c \psi \cdot t} \right) \]  

(21)

Consequently, the delay time \( \Delta \tau(t) \) of the correction process can be seen explicitly from the Equation.

### 4. Prediction and calculation based on a finite stochastic model with the real failure data

The finite stochastic model is very similar to the finite binomial distributed model of section 2, but has, due to the newly introduced correction processes, the most important property of continuous improvement of the critical error rate over time. The stochastic model is used for the determination of critical error distribution over a particular time period. Furthermore, the model serves as an estimation of the number of remaining critical errors [2]. The finite-stochastic model that is based on the correction process serves as an estimate of the additional time required for the improvement of a specific reliability.

Using the equations mentioned above for the detection and correction processes, interesting expectation values are obtained [2].

**Figure 4.1.** Detection- und Correction-Process.

Figure 4.1 shows a realistic and time curve of the detection and correction process. This process is based on real failure data. The effect of the time delay \( \Delta \tau(t) \) can be seen clearly in the correction.
process. During the so-called “learning phase” a critical number of errors \( \sum_{i=0}^{n} u_{t_i} = 5.6 \) are detected at a certain point in time \( t = 0.5 \cdot 10^4 h \), however, only \( \sum_{i=0}^{n} u_{t_i} = 3.08 \) critical errors are corrected. The repair time is relatively large. It can be seen that the repair time becomes smaller and smaller as time goes on. At the time \( t = 1.5 \cdot 10^4 h \), \( \sum_{i=0}^{n} u_{t_i} = 7.28 \) critical errors are detected. At the same time \( \sum_{i=0}^{n} u_{t_i} = 6.84 \), critical errors are corrected.

A realistic prognosis is obtained with the introduction of correction processes for critical errors. Conventional models cannot predict critical errors and secondly the models do not provide correction processes. Figure 4.2 shows conventional models on the left. It can be seen that neither the detection process nor the correction processes provide meaningful information. The conventional correction process fails, because almost zero errors have been corrected at time \( t = 0.5 \cdot 10^4 h \) (figure 4.2, picture 3). Even with the time progressing to time \( t = 2.5 \cdot 10^4 h \), no satisfying prognosis can be achieved.

The newly introduced method in figure 4.2, right, establishes a more reliable prognosis at time \( t = 0.5 \cdot 10^4 h \).

**Figure 4.2.** Detection- und Correction-Process.

**Conclusion**

It can be noted that conventional modes cannot describe a correction process for critical errors since only minimal failure data is available. Every convention model requires far more than the minimal amount of failure data to make reliable forecasts. This work presents a method that performs a prognosis of critical errors based on a correction process. The newly established correction process, however, requires that all errors are corrected completely.

**References**

[1] Musa, J.D.: Software Reliability Engineering, McGraw-Hill, New York, 1998.
[2] Huan-Jyh Shyur: A stochastic software reliability model with imperfect-debugging and change-point”, The Journal of systems and software, 2003.
[3] O.Krini, J.Börcsök: Different Approaches to Predict Software Reliability, Proceedings of 24th international congress, 2011.
[4] A. Birolini: Reliability Engineering: Theory and Practice”, Springer Verlag, Heidelberg, 2007.
[5] Lyu, M.: "Handbook of Software Reliability”, McGraw-Hill, New York, 1996