Improvement of Software Reliability Estimation Accuracy with Consideration of Failure Removal Effort

Myungmuk Kang
Department of Eco Vehicle Control Development, Hyundai Autron
688-1, Sampyeong-dong, Bundang-gu,Seongnam-si, Gyeonggi-do, Korea
E-mail: myungmuk.kang@hyundai-autron.com

Okjoo Choi
Department of Computer Science, KAIST,
291 Daehak-ro, Yuseong-gu, Daejeon, 305-701, Korea
E-mail: okjoo.choi@kaist.ac.kr

Juhwan Shin
Naval Combat System, Agency for Defense Development
P.O.Box 18, Jinhae, 645-600, Korea
E-mail: sharkshin@naver.com

Jongmoon Baik
Department of Computer Science, KAIST
291 Daehak-ro, Yuseong-gu, Daejeon, 305-701, Korea
E-mail: jbaik@kaist.ac.kr

Received 18 November 2012

Accepted 19 November 2012

In order to develop highly reliable software in a cost-effective manner, it is necessary to manage software reliability at the early test phases. Most of developers at those phases perform a test and debug activities together. In this paper, we propose a new reliability estimation model to manage the reliability of individual units from the early test phases as a solution for considering the test and debug time together. Via the proposed model using experiment of actual data, we can improve the accuracy of software reliability estimation.

Keywords: Software Reliability, Software Reliability Model, Exponential Model, Software Reliability Tool

1. Introduction

Software reliability is an important quality attribute that must be assured throughout a software development life cycle. The definitions of software reliability can vary according to the people and organizations involved. Based on the definition of the IEEE 1633 standard [3], software reliability is the ability of software to perform without failures under specified conditions during a specified time.

Therefore, the release time of software can be determined based on the estimated reliability. This implies that tests or releases of software will be determined according to the estimated number of residual software failures or the estimated time between software failures. Software reliability engineering focuses on engineering mechanisms for quantitative evaluations of software reliability, the development of software, and the maintenance of software. Thus, software reliability engineering refers to engineering techniques that quantitatively represent software behaviors according to users' requirements. Lyu [2] defined software reliability as an engineering process and stated that (1) reliability objectives, (2) operational profiles, (3) reliability modeling and measurement, and (4) reliability validation are more important than other

attributes. Particularly, the reliability modeling involves measuring and analyzing software reliability. In this paper, software reliability are measured and analyzed according to a novel model developed here.

Since 1970s, more than approximately 200 reliability models have been developed to estimate software reliability. These models have been applied to various domains, such as the medical, military, shipbuilding, and aerospace industry. Software reliability models have the form a mathematical expression that specifies the general form of the software failure process to estimate the initial failure counts, the time between failures, the remaining failures, and other factors based on detected failures, faults and the time that elapses during testing activities. Software reliability models are classified according to the phases of the software development life cycle [4, 5, 6]. Specifically, software reliability estimation models are classified into exponential models and S-shaped models [15]. These models are also classified into failure detection estimation models and failure removal estimation models [1]. Figure 1 shows the two exponential model and S-shaped model according to the test time.

In this paper, exponential software reliability models are used [4, 7] used during test phases. Specifically, in the Background section, three models are introduced to

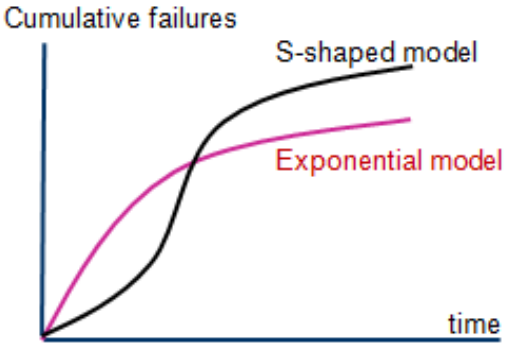


Fig. 1. Exponential Model and S-shaped Model

show that the proposed model represents a modification of exponential models considering characteristics that these models do not have. The Schneidewind model [3, 8], was developed to estimate total number of failures, remaining number of failures, and time to reach reliability goals based on the failure and time data collected during the test phases. Schneidewind also developed another estimation model to represent a failure removal process [9]. The model introduced the delay time, which is a different opinion from the assumption of the original Schneidewind model in which all failures are immediately removed. The integrated model of a failure detection and correction process [10] considers the delay time to correct failures and the failure detection process together. Existing models were developed to represent the failure detection process and correction process separately, as the test activities and debug activities are separately performed by developers and testers. However, at the early test phases and in small organizations, developers and testers are often not separated. This implies that developers should perform testing and debug activities together. Therefore, the proposed model was developed to support the situation that occurs frequently in the early test phases and in the small organizations.

2. Background

The exponential model is a basic model among software reliability models. The Goel-Okumoto model [4] is well known as a model of exponential models and the model proposed in this paper is also similar to the Goel-Okumoto model because the proposed model is developed through a modification of the Goel-Okumoto model. The Goel-Okumoto model is not described in this section, the Schneidewind model, a very well-known exponential model used to estimate failure detections, is described. Other models that estimate failure removal processes are also described.

2.1. *Failure detection estimation model*

One feature of the failure detection estimation models like the Schneidewind detection model [3, 8] is that it can also model failure detection processes. The Schneidewind detection model is a recommended model among various software reliability estimation models in the IEEE 1633 standard. It is validated based on the failure data of National Aeronautics and Space Administration (NASA) in the U.S. The Schneidewind detection model uses the detected failure counts within the same time interval and calculates the current failure rate based on the historical failure rate to predict future failures accurately. The Schneidewind detection model considers that the failure detection process can be changed when the test is performed and suggests a basic approach as well as two additional approaches considering that recent failure counts are more useful

than historical failure counts to predict near-future failures. It is possible to select one approach among the three approaches based on the purpose. According to each approach, parameter α and β can be estimated by the Maximum Likelihood Estimation (MLE)[11] method.

Three approaches of the Schneidewind model are as follows.

- Approach 1: use all of the failure counts from interval 1 through t (i.e., s = 1).
- Approach 2: use failure counts only in intervals s through t (i.e., $1 \leq s \leq t$).
- Approach 3: use cumulative failure counts in intervals 1 through s-1 and individual failure counts in intervals s through t (i.e., $2 \leq s \leq t$).

In order to use this model, it is necessary to follow the next process shown below.

- Assumptions for data collection
  - Perfect debugging
  - Removal time is ignored
- Data collection
  - Detected failures
  - Test time
- Estimation of parameters for the mean value function
  - Parameter for the total failure counts
  - Parameter from the failure occurrence rate
- Reliability validation
  - Estimation of failure counts that is undetected
  - Decision of the time point for test or release

2.2. *Failure removal estimation model*

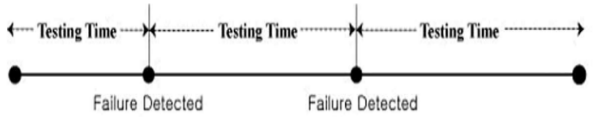
The failure removal estimation model is modeled from the failure removal process and the Schneidewind removal model and JungHua's removal model are representative models among failure removal estimation models [9, 10]. The Schneidewind removal model was developed from a modification of the basic Schneidewind detection model because the basic Schneidewind model has the unrealistic limitation in which the "removal time is ignored". Therefore, a delay time is introduced; this is the time between failure detection and removal. JungHua's removal model was developed from the basic Goel-Okumoto model. It considers the delay time and the failure correction rate. JungHua's removal model shows the failure removal process and considers the failure detection process and the removal process. According to these two functions, failure detection and correction processes can be estimated.

To use these models, it is necessary to follow the process shown below.

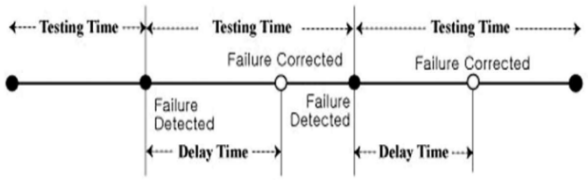
- Assumptions for data collection
  - Perfect debugging
  - Delay time occurs
- Data collection
  - Detected failures and removed failures
  - Test time and delay time
- Estimation of parameters for the mean value function
  - Parameter for total failure counts
  - Parameter from the failure occurrence rate and failure correction rate
- Reliability validation
  - Estimation of failure counts removed
  - Remaining uncorrected failure counts and time point of all detected failures removed

2.3. *Motivation*

Basic Non-Homogeneous Poisson Process (NHPP) models have the assumption that all detected failures are immediately removed and that it is possible to measure and analyze software reliability based on these assumptions. However, it is impossible to remove all failures immediately in reality. This signifies that a certain delay time must transpire [9, 10]. General estimation models do not consider the removal time of faults; only the test time is considered. This is a limitation of general estimation models. In order to deal with this limitation, improved models contain a new assumption. When a failure is detected, faults are removed after a certain time. However, in this paper, the definition of the delay time differs from that in the earlier research. According to the previous research, the delay time is the time between failure detection and failure correction. However, the removal time is different. The removal time is described as solely the time spent removing faults. Therefore, the delay time is larger than the removal time. Figure 2 describes the test time, delay time, and the removal time.



(a) Failure Detection Estimation Model



(b) Failure Removal Estimation Model

(c) Proposed Model

Fig. 2. Testing Time, Delay Time and Removal Time of each model

In the existing studies, the failure detection process and failure removal process are modeled based on the test time and delay time. This implies that general detection estimation models consider the test time under the limitation that the removal time is 0. General removal estimation models consider the test time to confirm failure detection and the delay time to confirm the failure removal process. As described above, existing models have a limitation in which the failure detection and removal processes are separated because existing models are normally used during system testing and operational testing in which developers and testers are normally separated, performing their test and debug activities independently.

However, developers and testers cannot be separated in the early test phases or in small organizations if performing testing and debug activities together. Therefore, failure detection and removal efforts cannot be separated. This indicates that a new estimation model to deal with this limitation must be used in the early test phases or in small organizations. Figure 3 shows software reliability models according to a software development life cycle. As shown in the figure 3, no suitable model exists for unit testing and integration testing. This paper suggests a feasible model for use in the test phases.

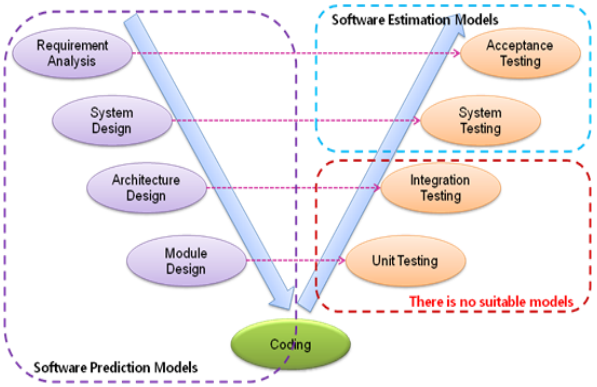


Fig. 3. Software Reliability Models on V-Model

3. A Reliability Estimation Model based on Software Fault Detection and Removal Effort

As described in the previous section, existing reliability estimation models are used based on testing and debug activities, which are performed separately. However, developers and testers are not separated during the early test phases and in small organizations . Developers perform tests to detect failures and debug the detected faults. Therefore, it is necessary to consider testing and debugging effort together. To support the idea, a new model is developed here. This section explains the approach of this research, including the description of the proposed model.

3.1. *Overview*

The proposed model in this paper is based on the Goel-Okumoto model [7], a well-known model among exponential models. Also the proposed model considers the characteristics of early test phases where developers create test cases and perform testing and debugging.

The Goel-Okumoto model has a Mean Value Function (MVF) to estimate cumulative detected failures.

$$MVF = \alpha \times [1 - \exp(-\beta t)] \quad \alpha > 0, \beta > 0$$

To use the MVF, it is necessary to estimate parameters α and β. The two parameters can be estimated by Maximum Likelihood Estimation (MLE) based on the collected data. Parameter α refers to the number of estimated total failures and the parameter β refers to the failure occurrence rate.

The Goel-Okumoto model estimates the total number of failures based only on test time and the number of detected failures. Therefore, estimating the total number of failures is straightforward. The proposed model also follows the number of total failures estimated by the Goel-Okumoto model. Thus, the two models show the same number of estimated total failures. However, the Goel-Okumoto model does not consider the failure removal time. Therefore, it is limited when estimating the accurate time when developers perform test and debug activities together at the early test phases because the Goel-Okumoto model can estimate detected failure counts based only on the test time. In order to deal with this limitation, it is necessary to change the failure rate according to the removal time. In other words, failure occurrence is delayed according to the removal time which then changes the failure occurrence rate. Two characteristics of the proposed model are as follows:

C1: the number of estimated total failures is identical between the basic Goel-Okumoto model and the proposed model.

C2: the failure occurrence rate is changed according to the removal time because detected failures occur at the delayed time point.

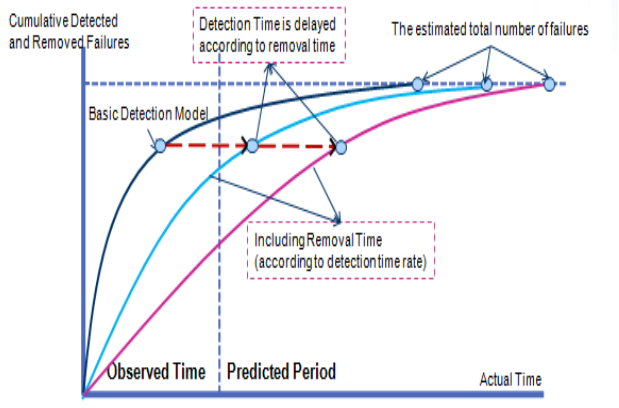


Fig. 4. Results of the Basic Exponential Model and the Proposed Model

Figure 4 shows the result of the basic exponential model; the graph changes when the removal time is included in the result of a basic exponential model. In this figure, all graphs have the same number of estimated total failures.

3.2. *Assumption*

The proposed model has the same assumptions as a basic exponential model because the proposed model is based on the exponential model. One of the representative assumptions is that all detected failures are immediately removed. Thus, the failure removal time is not considered and only the test time is considered. However, the removal time must be considered in actual development scenario. Therefore, new models have been developed to deal with the limitations [9, 10].

The proposed model also has different assumptions to consider failure the detection time and the removal time.

A1: all detected failures are immediately removed and a certain time is spent to remove detected faults.

A2: the failure occurrence rate is decreased at time goes on. Thus, the failure occurrence rate is decreased based on the time rate, which is the failure detection time divided by the total time.

In assumption A1, developers perform test activities and analyze detected failures to identify the cause of the failures to remove. Therefore, all detected failures are immediately removed. However, to remove detected failures, the removal time must be considered because developers must spend a certain amount of time. In assumption A2, if only the test time is considered, the failure occurrence rate will be identical to the basic exponential model. However, the failure occurrence rate must be decreased because the failure removal time is considered; the failure detection time rate is calculated by the total time including the test time and the debug time.

The proposed model was developed based on the assumptions of the basic exponential model and the two different assumptions are added. The two different assumptions are different from those in existing models and the characteristics of the unit and integration testing phases are considered to estimate a more accurate result.

3.3. *Proposed Model*

Below are the notations of the proposed model.

- α: the number of estimated total failures of the Goel-Okumoto model

- α_p: the number of estimated total failures of the proposed model (the same as α)
- **β**: failure occurrence rate per failure of the Goel-Okumoto model
- β_p: failure occurrence rate per failure of the proposed model
- **t**: failure detection time
- t_p: total time (failure detection time + failure removal time)
- **MVF (t)**: the cumulative number of detected failures between time 0 and time t of the Goel-Okumoto model.
- **MVF_p (t_p)**: the cumulative number of detected failures between time 0 and time t_p of the proposed model.

The proposed model uses the same procedure to collect data as used in existing models. Data regarding the detected failures during test activities is analyzed and the detected failure counts are rearranged in the same time interval. In addition, data concerning the faults that are cause of the failures is also collected because the proposed model considers the failure removal time. In the proposed model, the failure detection process is shown with total time as spent by developers. Therefore, the proposed model provides more accurate predictions of the time point of future failures.

The proposed model is based on the Goel-Okumoto model, which is a well-known basic exponential model. Therefore, the new notation MVFp is developed based on MVF in the Goel-Okumoto model. There also two new assumptions. Essentially, parameters α and αp, estimated according to the test time, are identical because the number of estimated total failures must not be changed.

$$P1: \alpha_p = \alpha \qquad (C1)$$

Parameter β of the failure occurrence rate, considering only the test time according to assumptions A1 and A2, is decreased according to the test time rate of the total time.

$$P2: \beta_p = \beta \times [T_d \times (T_d + T_r)] \quad (C2)$$

T_d is the time spent on detecting failures and T_r is the time spent on removing failures. Thus, the total time is the time to spend on detecting (T_d) and removing (T_r) failures. According to P1 and P2 above the basic MVF is changed to a new MVF to consider the failure removal time to predict cumulative detected failures.

$$MVF_p = \alpha_p \times [1 - \exp(-\beta_p t_p)]$$

As Figure 5 shows, it is necessary to follow certain steps to estimate the number of cumulative detected failures according to the proposed model. Estimation activities are that it is firstly necessary to collect failure-related data to estimate parameters α and β according to the equation as above, then parameters α_p and β_p are calculated from the new equations. The new MVF_p consists of new parameters α_p, β_p, and t_p to estimate the number of cumulative detected failures. Therefore, the number of cumulative detected failures is estimated based on fault detection and removal effort.

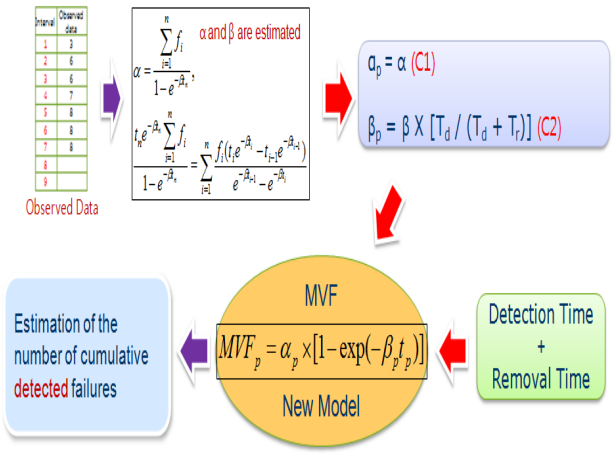


Fig. 5. Process of Proposed Model

To use the proposed model, it is necessary to follow the process below.

- Assumptions for data collection
  - Perfect debugging
  - No delay time
  - Removal time occurs
- Data collection
  - Detected failures and removed failures
  - Test time and removal time
- Estimation of the parameters of the basic Goel-Okumoto model
  - Parameter for total failure counts
  - Parameter from the failure occurrence rate
- Changes of parameters for the new MVF of the proposed model

- ▪ Parameter change to fit the proposed model
- ▪ Parameter α_p and β_p are calculated

• Calculation of the total time
- ▪ Failure detection time and removal time
- ▪ Calculation of total time

• Reset up parameters for the new MVF
- ▪ Parameter set up for α_p, β_p, and t_p
- ▪ Estimation of the result of the new MVF

• Reliability validation
- ▪ The number of cumulative failures and remaining failures
- ▪ Confirmation of current reliability

In section 2, the three existing models were introduced. These models are based on the exponential model and failure counts model. The proposed model was developed to improve the limitations of existing models. The result of the comparison is summarized in the table1. This table contains the phases, assumptions, removal time, integration of the detection time and the removal time, the outputs, and the usage of the models.

4. Validation

This section gives a validation of the proposed model to confirm how it fits actual data. To confirm whether the proposed model can be used without problems, collected industrial data was used during the early test phases of the unit and integration testing phases. To validate the proposed model, a proper environment was set up and an experiment conducted.

4.1. *Environment for the experiment*

There are two existing open tools, CASRE and SMERFS, to estimate software reliability with models based on data collected in a convenient manner [12, 13]. However, it is very difficult to compare the results of the Goel-Okumoto model and the proposed model with these two open tools. To deal with this difficulty, a new software reliability analysis tool developed by us, known as SRTpro [14], was used for the experiment. Also SRTpro was developed to deal with the limitations and drawbacks of existing tools. Figure 6 shows a screenshot of SRTpro.

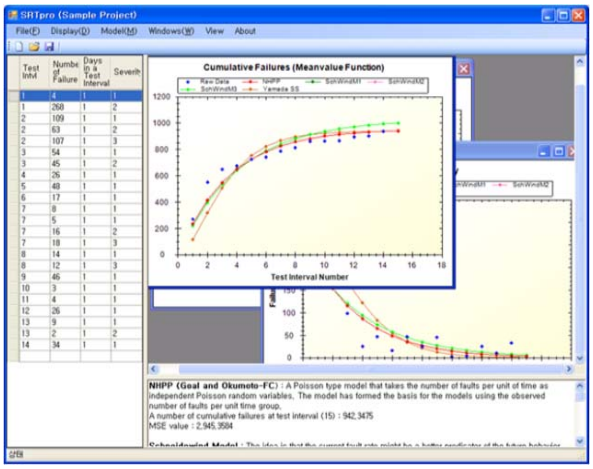


Fig. 6. Screenshot of SRTpro

4.2. *Data Collection*

To conduct an experiment of the proposed model, actual data collected from an organization was used instead of hypothetical data. The actual data was collected from the in-progress project of a CMMI level 5. To collect the actual data, the two data collection templates were used, as shown in Figure 7 and Figure 8 below. Developers collected the actual data using forms and sent them to us.

Figure 7 shows the template used to collect detected failures during test activities, and Figure 8 shows the template used to collect faults which were the causes of the failures. The removal time is included. All experimental data was collected using the data collection templates and the data was rearranged for use as input data in the reliability models.

Milestone	CSCI	CSC	Detection Date	Test Script ID	Test Cases	People	Time	cpu time	Total Time	Failures	Faults
Integration testing	A	A-1	2009-12-08	1	10	1	215	105	215	2	3

Fig. 7. Failure Collection Template

Milestone	CSCI	CSC	Fault ID	Test Script ID	i th test case	Removal date	Severity	People	Removal time
Integration testing	A	A-1	1	1	6	2009-12-08	Major	1	50
			2	1	7	2009-12-09	Minor	1	50

Fig. 8. Fault Collection Template

The actual data was collected in 28 files, and only 10 CSCs were selected for the experiment because unusable CSCs had problems such as not being fitted to an exponential model, CSCs that were too small, no failures detected, and too short a test time. However, more CSCs can be used for the experiment through more analysis activities of the actual data.

4.3. *Goals of experiment*

The primary goal of the experiment was to confirm relationship between the result from the proposed model and the actual collected data. First, it sought to confirm the limitation that the Goel-Okumoto model, which does not consider the time to remove faults according to the difference between the estimation results of the Goel-Okumoto model based on the collected data without the removal time and the actual data including the removal time. Second, it sought to confirm how to improve the limitation according to the difference between the estimation results of the proposed model and the actual data including the removal time. To compare estimation results to actual data, Mean Relative Error (MRE) [10] and Mean Square Error (MSE) [10] mechanisms were used to confirm the difference. The scope of the values was confirmed through a box-Plot between the actual collected values and the estimated values of the Goel-Okumoto model and the proposed model.

Mean Relative Error (MRE)

$$\frac{1}{n}\sum_{k=1}^{n}\left|\frac{m(t_k)-Z_k}{Z_k}\right|$$

Z_k is the number of failures occurring at the time interval k as observed during test the activities. $m(t_k)$ is the number of cumulative failures at the time point k as estimated through MVF. n is the number of total time intervals. It was used to confirm the difference between the estimation results and the actual values; when the MRE value is small, it can be interpreted that there is no difference between the estimation results and the actual values, implying that it is possible to estimate the number of future failures more accurately.

Mean Square Error (MSE)

$$\frac{1}{n}\sum_{k=1}^{1}\left|m(t_k)-Z_k\right|^2$$

Table 1. Comparing of Existing Models and the Proposed Model

	Detection model	Removal Model	Proposed Model
Phases	• System testing • Operational testing		• Unit and Integration testing
Assumptions	• Perfect debugging • No delay time • No removal time	• Perfect debugging • Delay time for removal	• Perfect debugging • No delay time, but removal time exists
Removal Time	X	Δ (delay time)	O
Integration of detection time and removal time	X	Δ (separately use)	O
Outputs	• The number of cumulative detected failures • The number of remaining undetected failures	• The number of cumulative removed failures • The number of remaining unresolved failures	• The number of cumulative detected failures • The number of remaining undetected failures
Usage	• To decide when to stop testing	• To know the time when to finish debugging failures	• To manage reliability at the early phases • To decide when to move on the next step

Z_k is the number of observed failures occurring at the time interval k and $m(t_k)$ is the number of cumulative failures at the time interval k as estimated through MVF, as above. A MSE value is the average value of the squared value of the difference between the two values. It is used to confirm the difference between the estimation values and the actual values. In most cases to determine the priorities of several models based on collected data, MSE values are normally used. If the MSE value is small, it indicates that the estimation values are close to the actual values and that it is possible to estimate the number of future failures more accurately.

The next section describes the confirmation results of the experiment. To confirm that there exist a difference between the estimation values of the proposed model and the actual values statistically according to Paired T Test, the tool known as "Minitab" is used.

4.4. *Experiment and Validation Result*

To conduct the experiment, the collected failure data of a unit among 10 available units was used. The experiment produced several graphs of the results from the Goel-Okumoto and from the proposed model for validation. Shown first are the MVF result of the Goel-Okumoto model, ignoring the removal time in Figure 9.

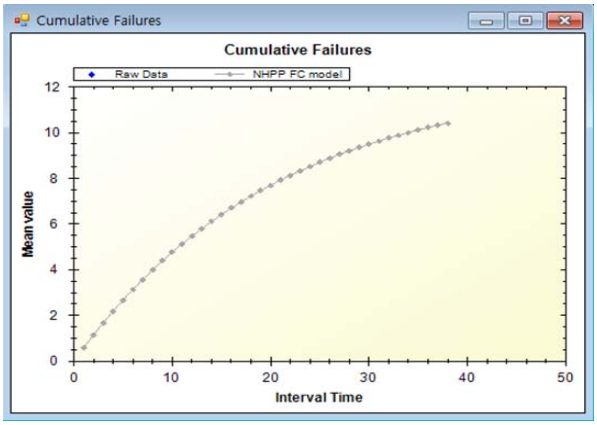


Fig. 9. Estimation Values of Goel-Okumoto Model

Figure 9 shows a graph of the cumulative failures estimated through MVF considering the failure counts and the test time. 10 failures are estimated at the time interval of 34 and 10.41 failures are estimated at the time interval of 38 in the graph. The next figure describes a comparison of the results of the proposed model considering the removal time of failures and results of the Goel-Okumoto model together.

Developers essentially spend a certain amount of time finding failures. They also spend a certain amount of time to remove faults that are the causes of these failures. In this experiment developers spent 120 to remove all of the faults. Therefore, the proposed model considers the removal time to reflect delay, as shown in the figure.

Developers essentially spend a certain amount of time finding failures. They also spend a certain amount of time to remove faults that are the causes of these failures. In this experiment developers spent 120 to remove all of the faults. Therefore, the proposed model considers the removal time to reflect delay, as shown in the figure.

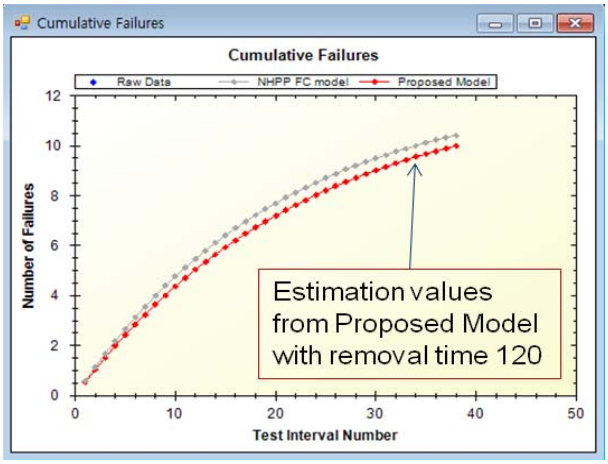


Fig. 10. Estimation Values of the Goel-Okumoto Model and the Proposed Model

The next figure shows two graphs: a graph of the Goel-Okumoto model and a graph of the proposed model with actual data including the test and debug time required by developers. As Shown in this figure, the proposed model considering the removal time of all faults is more accurate than that in the Goel-Okumoto model considering only the test time.

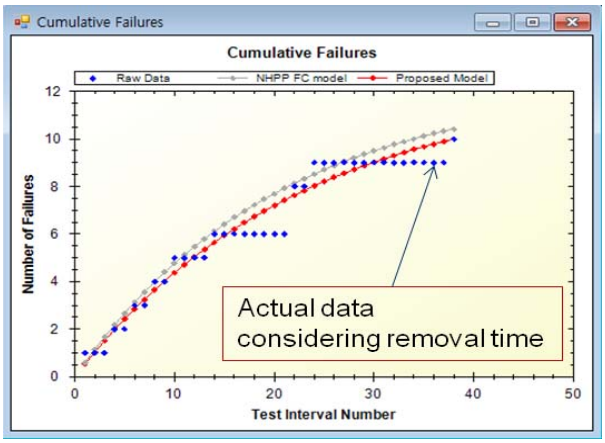


Fig. 11. Estimation Values and Actual Collected Values

The table below gives a comparison of the Goel-Okumoto model and the proposed model with estimated values according to the actual collected data. Thus, in the early test phases, developers normally perform testing and debugging activities, but the Goel-Okumoto model only reflects the test activities. Therefore, there is an error between the estimated values and the actual collected values.

Table 2. Description of Measurement of the Goel-Okumoto and the Proposed Model

	Goel-Okumoto Model	Proposed Model
Time	1140	1140
Detection time rate	-	0.8947
α	12.3437	12.3437
β	0.0489	0.0437
MRE	0.1276	0.0909
MSE	0.5964	0.2990
Total intervals	38	38
Failures	10.4161	10

From this table, it is clear that the Goel-Okumoto model has larger MRE and MSE values because the model does not have the assumption that developers perform testing and debugging activities in the early test phases. As mentioned earlier, smaller MRE and MSE values indicate that the model, with its smaller MRE and MSE values, is more accurate to estimate future failures because the current failure behavior shows a better fit to the actual collected data. As shown in the table, the proposed model has smaller MRE and MSE values than those of the Goel-Okumoto model. The MRE value of the proposed model is 0.0909 and the MRE value of the Goel-Okumoto model is 0.1276. The MSE value of the proposed model is 0.2990 while the MSE value of Goel-Okumoto model is 0.5964.

The difference between the estimated values and the actual values were determined through a box-plot. Figure 12 shows the result of the box-plot, demonstrating the difference. As shown in this figure, the medium value of the proposed model is close to 0, whereas the medium value of the Goel-Okumoto model is not close to 0 because the removal time of the faults is not considered.

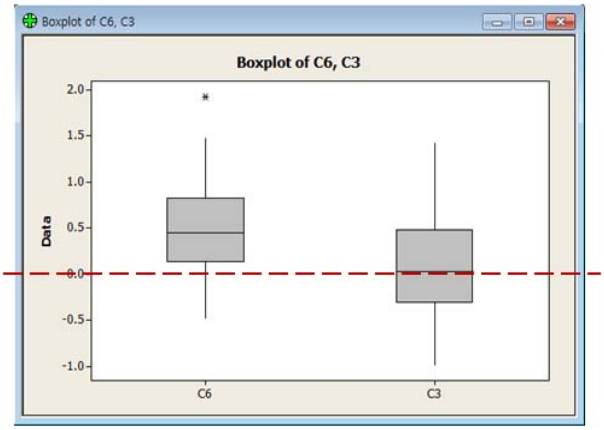


Fig. 12. Results of a Box-plot of the Goel-Okumoto and the Proposed Model

In Table 3, the measured values from the box-plot from Figure 12 are listed. Through Table 3, the difference between the medium values is identified.

Table 3. Measurements of the Box-plot

	Goel Model (C6)	Proposed Model (C3)
Max	0.8236	0.4844
Min	-0.1321	-0.3290
Median	0.4463	0.0260
IQRange	0.6915	0.7873
N	38	38

An experiment was conducted and the results described to explain the difference between the Goel-Okumoto model and the proposed model thus far. The differences between the two models were clearly identified from the experiment. When developers perform testing and debugging activities, the estimated values of the proposed model are closer to the actual values compared to those of the Goel-Okumoto model. Table 4 and Table 5 show results of a Paired T Test done to confirm that the difference is statistically significant. In order to use a Paired T Test, it is necessary to define a null hypothesis and an alternative hypothesis. The bull hypothesis is that there is no significant difference and the alternative hypothesis is that there is a significant statistical difference.

Table 4. Goel-Okumoto Model and the Proposed Model in Paired T Test

	N	Mean	StDev	SE Mean
Goel	38	0.501650	0.537837	0.087249
Proposed	38	0.113082	0.550028	0.089225
Difference	38	0.388568	0.186429	0.030243
T-Value	12.85		P-Value	0.000

First, a Paired T Test was conducted to confirm the statistical difference between the difference between the actual values and the estimated values of the Goel-Okumoto model and the difference between the actual values and the estimated values of the proposed model. From Table 4, the P value of the result of the two models in 95% significant level is 0, which is obviously smaller than 0.05. Therefore, the null hypothesis is rejected and the alternative hypothesis is selected. This implies that there is a significant difference between the two models.

Second, another Paired T Test was conducted to confirm the statistical difference between the actual values and the values estimated by the proposed model. From Table 5, P value of the result of the proposed model at the 95% significant level is 0.213, which is larger than 0.05. Therefore, the null hypothesis is not rejected. This indicates that the alternative hypothesis is not selected, implying that there is no evidence to explain the significant difference between the actual values and the estimated values of the proposed model. Therefore, it is possible that there is no significant difference.

Table 5. Actual Collected Data and the Proposed Model in Paired T Test

	N	Mean	StDev	SE Mean
Actual Data	38	6.34211	2.76343	0.44829
Estimated Data from proposed model	38	6.645519	2.79828	0.45394
Difference	38	-0.113082	0.550018	0.089225
T-Value	-1.27		P-Value	0.213

This section described the experiment and results to confirm the differences between the two models. Also described are the results of the Paired T Test to confirm whether is a significant difference exists statically. The experiment identified that the proposed model has more accurate estimation results when developers perform testing and debugging activities together. It was also found that there is a significant difference between the Goel-Okumoto model and the proposed model according to statistical validation methods. It was also whether a statistically significant difference exists between the actual values and the values estimated by the proposed model.

5. Conclusions

In this paper, a new reliability estimation model was developed to consider the characteristics of the early test phases. Current existing reliability estimation models are normally used during the late test phases, which typically include system testing and operational testing. Therefore, the current existing reliability estimation models can be divided into failure detection estimation models and failure removal estimation models.

Failure detection estimation models consider the test time to estimate future failure trends using the detected failure counts per time interval. Failure removal estimation models estimate that future failure will be removed using the removed failures detected per time interval. These models do not consider that developers perform testing and debugging activities

together during the early test phases because the models are used in the late test phases and because developers and testers are separated in the late test phases.

To deal with these limitations, a new model was proposed that considers the test time and the debug time together, making it possible to manage software reliability from the early test phases to the late test phases through the proposed model. Therefore, the failure removal cost will be reduced.

An experiment was conducted to compare the existing model and the proposed model with actually collected industrial data, and a Paired T Test was used to confirm the difference between the proposed model and the existing model and between the actual collected values and the estimated values by the proposed model. This experiment confirmed whether there was a statistically significant difference.

Through the proposed model, it was possible to provide more accurate estimation results on the test side. Moreover, the failure removal cost during the late test phases can be reduced on the development side. On the management side, the three steps of the reliability process, the software reliability prediction models, early estimation models, and late estimation models, can supported while reducing the schedule management cost as well as the total cost. On the project side, the high possibility of software development is enhanced.

Currently, the proposed model is based on Exponential models. Therefore, data that is fitted to S-shaped models cannot be used with the proposed model. This is a limitation of the proposed model. To deal with this limitation, the proposed model will be expanded to use data that is fitted to S-shaped models. The expanded model will provide MRE and MSE values to select more accurate models according to the estimated results. Additional experiments will also be conducted to confirm accuracy of the model based on data collected during integration testing to provide more accurate evidence for use with the model.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0018020).

References

1. Reliability Analysis Center, “Introduction to Software Reliability: a State of the Art Review”, Rome Laboratory, 1996
2. Michael R. Lyu, “Software Reliability Engineering: A Roadmap”, FOSE, 2007.
3. “IEEE Recommend Practice on Software Reliability”, IEEE Reliability Society, June, 2008.
4. M. Xie, “Software Reliability Modeling”, WorldScientific, 1991.
5. Ch. Ali Asad Muhammad Irfan Ullah, Muhammad Jaffar-Ur Rehman, “An Approach for Software Reliability Model Selection”, Computer Software and Applications Conference, 2004.
6. Yinong Chen and Jean Arlat, “An Input Domain-Based Reliability Growth Model and Its Applications in Comparing Software Testing Strategies”, LAAS REPORT, April, 1995.
7. Michael R. Lyu, “Handbook of Software Reliability Engineering”, IEEE Computer Society Press, 1997..
8. Norman F. Schneidewind, “Reliability Modeling for Safety-Critical Software”, IEEE Transactions on Reliability, March, 1997.
9. Norman F. Schneidewind, “Modeling the Fault Correction Process”, 12th International Symposium on software Reliability, November, 2001.
10. Jung-Hua Lo, Chin-Yu Huang, “An Integration of Fault Detection and Correction Processes in Software Reliability Analysis”, The Journal of Systems and Software, 2006.
11. Robert V. Hogg, Joseph W. McKean, Allen T. Craig, “Introduction to Mathematical Statistics”, Pearson, 2005.
12. Allen Nikora, “CASRE-A Computer-Aided Software Reliability Estimation Tool”, http://www.openchannelfoundation.org/projects/CASRE_3.0.
13. William Farr, Oliver Smith, “SMERFS-Statistical Modeling and Estimation of Reliability Functions for Systems”, 1996.
14. Myungmuk Kang, Taewan Gu, Jongmoon Baik, “A User Friendly software Reliability Analysis Tool based on Development Process to Iteratively Manage Software Reliability”, International Symposium on Software Reliability Engineering, 2009.
15. S. Yamada, M. Ohba, O. Osaki, “S-Shaped Reliability Growth Modeling for Software Error Detection, IEEE Transactions on Reliability”, Vol, R-32, no. 5475-5478, 1983.