Software Optimal Release Problem Considering the Environment for the Usage of Mobile Device

YOSHI NOBU TAMURA1 and SHIGERU YAMADA2
1Graduate School of Science and Engineering, Yamaguchi University, JAPAN
2Graduate School of Engineering, Tottori University, JAPAN
(Received on April 24, 2013, revised on April 21, 2013)

Abstract: An embedded OSS known as one of OSS’s has since been gaining a lot of attention in embedded system area such as Android, BusyBox, etc. However, the poor handling of quality and customer support prohibit the progress of embedded OSS. In this paper, we propose a method of software reliability assessment based on a flexible hazard rate model considering the influence of installer application. Also, we derive several reliability assessment measures. In particular, we show several numerical examples of reliability assessment based on our hazard rate model. Moreover, we formulate a total expected software cost model considering the environment for the usage of mobile device. Then, we discuss about the determination of optimum software release times minimizing the total expected software cost.

Keywords: Imbedded software, reliability, hazard rate model, mobile device, optimal release problem.

1. Introduction

OSS (Open Source Software) systems which serve as key components of critical infrastructures in the society are still ever-expanding now. At present, the mainstream of software development environment is the development paradigms such as a concurrent distributed development environment. The open source project contains special features so-called software composition by which several geographically-dispersed components are developed in all parts of the world. The successful experience of adopting such open source projects includes Apache HTTP server[1], Firefox Web browser[2], and GNU/Linux operating system. Especially, an embedded OSS known as one of OSS has been gaining a lot of attention in embedded system area, i.e., Android, BusyBox, etc., because of the cost reduction, quick delivery, and work saving. However, the poor handling of quality and customer support prohibit the progress of embedded OSS.

In particular, as one of software reliability growth models (SRGM's)[3-5], hazard rate models[6-9] have been applied to assess the reliability for quality management and testing-progress control of software development. Also, several methods of dynamic testing management for new distributed development paradigms as typified by the open source project have been presented[10-12]. In case of considering the effect of the debugging process on entire system in the development of a method of reliability assessment for OSS, it is necessary to grasp the situation of registration for the bug tracking system, the degree of maturation of OSS, and so on. However, it is difficult for developers to assess the reliability and portability of embedded OSS on a single-board computer. The term “porting-phase” means the rebuilding process in which the developers create an OS/application developed for the specific computer system to suit another computer system. From above mentioned problems, many companies have been hesitant to innovate the embedded OSS.

*Corresponding author’s email: tamura@yamaguchi-u.ac.jp
Many fault-counting type SRGM’s have been applied to assess the reliability for quality management and testing-progress control of software development. However, it is difficult to apply the SRGM's to the OSS, because the number of detected faults in the OSS project can not converge to a finite value [13,14]. In fact, there are several SRGM's that can be applied in the above discussed situation, i.e., the Weibull and Log-logistic SRGM’s, and so on [3]. In particular, in case that the number of detected faults can not converge to a finite value, it is difficult to assess whether the porting phase will succeed by using reliability assessment measures derived from SRGM’s. Also, the hazard rate models have the simple structure.

In this paper, we focus on the problems in software quality, which prohibit the progress of embedded OSS. We propose a method of software reliability assessment based on the flexible hazard rate model considering the influence of installer application such as App Store of iOS and Android Market of Android OS. Also, we derive several reliability assessment measures. In particular, we show several numerical results of reliability assessment for our hazard rate model. Moreover, it is very important in terms of software management that we decide for the optimal length of the porting-phase for embedded OSS. We find the optimum release time of porting-phase by minimizing the total expected software cost. Then, we formulate a total expected software cost model based on our proposed hazard rate model for the embedded OSS considering the environment for the usage of mobile device. Then, we also show that the proposed model can assist quality improvement for embedded OSS systems development.

## 2 Model Description

The time-interval between successive faults of \((k-1)\) st and \(k\) th is represented as random variable \(X_k\) \((k = 1, 2, \cdots)\). Therefore, we can define the hazard rate function \(z_k(x)\) for \(X_k\) as follows:

\[
z_k(x) = w_k(x)N - (k - 1)\quad (k = 1, 2, \cdots, N; N > 0), \quad (1)
\]

\[
w_k(x) = \phi \cdot \exp\left[-\frac{k}{\alpha}\right] \quad (k = 1, 2, \cdots, N; \phi > 0, \alpha > 0), \quad (2)
\]

where the notations in Eqs.(1) and (2) are represented as follows:

- \(z_k(x)\): the hazard rate for \(X_k\) \((k = 1, 2, \cdots)\),
- \(N\): the number of latent faults in embedded OSS,
- \(\phi\): the hazard rate per inherent fault for embedded OSS,
- \(\alpha\): the number of latent application softwares installed by using the installer software.

Eq.(2) means the hazard rate per inherent fault for embedded OSS changes with increasing the number of application softwares by using installer application. The proposed model has \(\alpha\) as the unique parameter. The proposed model can cover both the trends of reliability growth and reliability regression according to the value of parameter \(\alpha\). Thereby, the proposed model can assess the trend of reliability regression in the porting phase.

## 3 Reliability Assessment Measures

In porting-phase of embedded OSS, the distribution function of \(X_k\) \((k = 1, 2, \cdots)\) representing the time-interval between successive faults of \((k-1)\) st and \(k\) th is defined as:
where $\text{Pr}\{A\}$ represents the occurrence probability of event A. Therefore, the following derived function means the probability density function of $X_k$:

$$f_k(x) = \frac{dF_k(x)}{dx}.$$  

(4)

Also, the software reliability can be defined as the probability which a software failure does not occur during the time-interval $(0, x]$ after the porting-phase. The software reliability is given by

$$R_k(x) = \text{Pr}\{X_k > x\} = 1 - F_k(x).$$  

(5)

From Eqs.(3) and (4), the hazard rate is given by the following equation:

$$z_k(x) = \frac{f_k(x)}{1 - F_k(x)} = \frac{f_k(x)}{R_k(x)},$$  

(6)

where the hazard rate means the software failure rate after the porting-phase when the software failure does not occur during the time-interval $(0, x]$.  

4 Numerical Examples of Reliability Assessment

There are many open source projects around the world. In particular, we focus on the embedded OSS in order to evaluate the performance of our method, i.e., Android[15] and BusyBox[16]. BusyBox includes 4 estimate components. In this section, we focus on the Android OS developed for mobile phone. In particular, we consider the case of installing BusyBox to Android as the porting environment. We use the actual data collected from the following versions.

- Android 1.5 NDK, Release 1
- Android 1.1 SDK, Release 1
- Android 1.0 SDK Release 1 available

Considering the realities of the field use, we show the performance illustrations of our tool by using the data sets in terms of “Android 1.0 SDK Release 1” and “Android 1.5 NDK”. For one example, Android 1.0 SDK Release 1 is the stable version because of the compact package and the first major release version. On the other hand, Android 1.5 NDK is the unstable version because of the large size of source code and the complexity of system. Thus, we illustrate the method of reliability assessment for the porting-phase assuming the above-mentioned porting environment.

We show the estimated MTBF for each version in Figures 1~3. From Figures 1~3, we can find that the estimated MTBF shows increase very little in case of Android 1.1 SDK. Figure 2 implies the trend of reliability regression.
Optimal Software Release Problem for the Porting Phase

Recently, it becomes more difficult for software developers to produce highly-reliable software systems efficiently, because of the more diversified and complicated software requirements. Thus, it has been necessary to control the software development process in terms of reliability, cost, and delivery time [17,18]. Especially, it is difficult for software developers to manage the porting-phase of the embedded system development by using the embedded OSS. Also, it is very important in terms of software management that we decide for the optimal length of the porting-phase for embedded OSS. We find the optimum release time of porting-phase based on the total expected software cost in this section. We formulate a maintenance cost model based on our proposed hazard rate model for embedded OSS. It is interesting for the software developers to predict and estimate the optimum time when we should stop bug fixing in order to develop a highly reliable software system efficiently. Then, we discuss about the determination of optimum software release times minimizing the total expected software cost considering the environment for the usage of mobile device.

The followings are defined:

- $c_1$: the testing cost per porting-time ($c_1 > 0$),
- $c_2$: the fixing cost per fault during the porting-phase ($c_2 > 0$),
- $c_3$: the fixing cost per fault after the release ($c_1 > c_2$).

Then, the expected software cost of OSS can be formulated as:
\[ C_i(l) = c_1 \sum_{k=1}^{l} E[X_k] + c_2 l, \]  
(10)

where \( l \) is the number of \( l \)-th software failure occurrence. Also, the expected software maintenance cost after the release is represented as follows:

\[ C_2(I) = c_3 (p \cdot \alpha + N - l). \]  
(11)

where \( p (p > 0) \) is the parameter in terms of the environmental factor for the usage of mobile device after the release. Consequently, from Eqs.(10) and (11), the total expected software cost is given by

\[ C(I) = C_1(I) + C_2(I). \]  
(12)

From \( I^{*} \) obtained by minimizing \( C(I) \), we can estimate the optimum software release time as \( \sum_{k=1}^{I^{*}} E[X_k] \).

6 Numerical Examples for the Optimal Release Time

We show numerical examples for the optimum release time of the porting phase by using Android[15] and BusyBox[16]. Figures 6–8 shows the estimated total expected software cost where \( c_1 = 1, c_2 = 2, c_3 = 3, p = 1 \).

For example, from Figure 4, we find that the estimated number of fault detection \( I^{*} \) which minimizes the estimated total expected software cost is 908.23. Then, the optimum software release time \( I^{*} \left( \sum_{k=1}^{908} E[X_k] \right) \) is 14.445 days. On the other hand, “Android 1.1 SDK, Release 1” shows that the estimated total expected software cost continues to decrease because of the trend of reliability regression. Similarly, from Figure 5, we find that the estimated number of fault detection \( I^{*} \) which minimizes the estimated total expected software cost is 495.18. Then, the optimum software release time \( I^{*} \left( \sum_{k=1}^{495} E[X_k] \right) \) is 6.2787 days.

![Figure 5: The estimated total expected software cost (Android 1.0 SDK, Release 1).](image1)

![Figure 6: The sensitivity analysis of parameter \( p \) in terms of total expected software cost (Android 1.5 NDK, Release 1).](image2)
7 Sensitivity Analysis in terms of the Environment for the Usage of Mobile Device

We have verified that our software cost model can be applied to assess quantitatively optimum software release time in the porting-phase of the embedded system development. In this section, we show some behavior of the total software cost if we change the parameters $p$ included our software cost model. Figures 6~8 show the estimated total expected software costs in case of changing the parameter $p$. From the above results, if the environmental factor for the usage of mobile device is large, the estimated total expected software cost and optimum release time become large. This means that the total software cost and optimum release time increase with increase in the environmental factor for the usage of mobile device, i.e., the environmental factor depends on the operating time, frequency in use, the number of installations, and the number of users of mobile device, etc. According to the changes in the usage situation of mobile device, it is important for software managers to make an adjustment to the period of porting phase. By using the parameter $p$ of our software cost model, it is possible to assess the optimum release time and total expected software cost the considering the usage situation of mobile device.

8 Concluding Remarks

In this paper, we have discussed the method of software reliability assessment based on the flexible hazard rate model considering the installer application. In particular, we have assumed that embedded system includes an installer application in the porting-phase of embedded OSS. Also, we have derived several assessment measures based on our hazard rate model. By using our flexible hazard rate model, we can incorporate the complicated situation of the embedded system applied OSS, the degree of maturation of OSS, several software applications, and the installer application, etc.

Furthermore, we have formulated the total software cost considering the environment for the usage of mobile device by using our model. We have found that our method can evaluate the optimum software release time in the porting-phase of the embedded system development by applying embedded OSS.

Finally, we have focused on an embedded OSS developed under open source projects. Our software cost model would be useful as the method of reliability assessment for embedded OSS considering the environment for the usage of mobile device.

Acknowledgements: This work was supported in part by the Grant-in-Aid for Scientific
Research (C), Grant No. 24500066 and No. 25350445 from the Ministry of Education, Culture, Sports, Science, and Technology of Japan.

References


Yoshinobu Tamura received the B.S.E., M.S., and Ph.D. degrees from Tottori University in 1998, 2000, and 2003, respectively. From 2003 to 2006, he was a Research Assistant at Tottori University of Environmental Studies. From 2006 to 2009, he was a Lecturer and Associate Professor at Faculty of Applied Information Science of Hiroshima Institute of Technology, Hiroshima, Japan. Since 2009, he has been working as a Associate Professor at the Graduate School of Science and Engineering, Yamaguchi University, Ube, Japan. His research interests include reliability assessment for open source software. He is a regular member of the Information Processing Society of Japan, the Operations Research Society of Japan, the Society of Project Management of Japan, the IEEE.
Dr. Tamura received the IEEE Reliability Society Japan Chapter Awards in 2007 and the Research Leadership Award in Area of Reliability from the ICRITO in 2010, and the Best Paper Award of the IEEE International Conference on Industrial Engineering and Engineering Management in 2012.

**Shigeru Yamada** received the B.S.E., M.S., and Ph.D. degrees from Hiroshima University, Japan, in 1975, 1977, and 1985, respectively. Since 1993, he has been working as a professor at the Department of Social Management Engineering, Graduate School of Engineering, Tottori University, Tottori-shi, Japan.

He has published over 500 reviewed technical papers in the area of software reliability engineering, project management, reliability engineering, and quality control. He has authored several books entitled such as Introduction to Software Management Model (Kyoritsu Shuppan, 1993), Software Reliability Models: Fundamentals and Applications (JUSE, Tokyo, 1994), Statistical Quality Control for TQM (Corona Publishing, Tokyo, 1998), Software Reliability: Model, Tool, Management (The Society of Project Management, 2004), Quality-Oriented Software Management (Morikita Shuppan, 2007), Elements of Software Reliability – Modeling Approach- (Kyoritsu Shuppan, 2011), and Project Management (Kyoritsu Shuppan, 2012).

Dr. Yamada received the Best Author Award from the Information Processing Society of Japan in 1992, the TELECOM System Technology Award from the Telecommunications Advancement Foundation in 1993, the Best Paper Award from the Reliability Engineering Association of Japan in 1999, the International Leadership Award in Reliability Enng. Research from the ICQRIT/SREQOM in 2003, the Best Paper Award at the 2004 International Computer Symposium, the Best Paper Award from the Society of Project Management in 2006, the Leadership Award from the ISSAT in 2007, the Outstanding Paper Award at the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM208) in 2008, the International Leadership and Pioneering Research Award in Software Reliability Engineering from the SREQOM/ICQRIT in 2009, the Exceptional International Leadership and Contribution Award in Software Reliability at the ICRITO’2010, and 2011 Best Paper Award from the IEEE Reliability Society Japan Chapter in 2012.

He is a regular member of the IEICE, the Information Processing Society of Japan, the Operations Research Society of Japan, the Japan SIAM, the Reliability Engineering Association of Japan, Japan Industrial Management Association, the Japanese Society for Quality Control, the Society of Project Management, and the IEEE.