Performability Analysis Considering Debugging Behaviors for Open Source Solution

YOSHINOBU TAMURA¹ and SHIGERU YAMADA²

¹Graduate School of Science and Engineering, Yamaguchi University, Japan
²Graduate School of Engineering, Tottori University, Japan

(Received on February 16, 2012, revised on September 24, 2012)

Abstract: A large scale open source solution composed of several open source softwares is now attracting attention as the next-generation software development paradigm because of the cost reduction, quick delivery, and work saving. The testing phase of open source solution means the binding testing and system testing. Generally, it makes no sense to perform the unit testing of open source solution, because the unit testing means the test for open source software. Then, it is very important for software managers to assess the binding status of open source solution. In this paper, we propose a new approach to software performability analysis based on stochastic differential equations in order to consider an interesting aspect of the collision status during the binding-phase of open source softwares in open source solution. Especially, we derive several performability assessment measures from our stochastic differential equation model. Also, we analyze actual software fault-count data to show numerical examples of software performability assessment considering the component collision for several open source softwares. Moreover, we show that the proposed performability analysis can assist improvement of quality for the large scale open source solution.

Keywords: Performability, reliability, open source software/solution, stochastic differential equations.

1. Introduction

Open source software (OSS) systems which serve as key components of critical infrastructures in the society are still ever-expanding now. The successful experience of adopting OSS includes GNU/Linux operating system, Apache HTTP server, and so on[1]. However, the poor handling of the quality and customer support prohibits the progress of OSS. This paper focuses on the problems of software quality, which prohibit the progress of OSS. Many OSS's are used because of the cost reduction, quick delivery, and work saving. Also, the development cycle of OSS has been repeated without a break. It is important for software project managers to assess the reliability of open source solution developed under several OSS's. At present, a large scale open source solution by using several OSS's is now attracting attention as the next-generation software development paradigm because of the cost reduction, quick delivery, and work saving. Also, the large scale open source solution has such a unique feature that it is composed of several OSS's. For many OSS's, there are no testing phases for their development cycles. However, the open source solution has the testing phase such as the binding phase of OSS's with debugging process. Then, it is very important to be able to connect several open source components. Also, it is need to consider the irregular fluctuation associated with the early binding phase of open source solution. Thereby, the software project managers can consider the characteristics of open source solution. 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 bug tracking system, the combination status of OSS’s, the degree of maturation of OSS, etc. Moreover, if the size of the software system is large, the number of faults detected during the operating phase becomes large, and the change of the number of faults which are detected and removed through each debugging becomes sufficiently small compared with the initial fault content at the beginning of operation. Therefore, in such a case, it is appropriate to use a stochastic model with continuous state space in order to describe the stochastic behavior of the fault-detection process such as the open source solution. Many software reliability growth models (SRGM’s)[2] have been applied to assess the reliability for quality management and testing-progress control of software development. On the other hand, the effective method of dynamic debugging management for new distributed development paradigm as typified by the open source project has only a few presented[3-6].

In this paper, we focus on an open source solution developed under several OSS’s. We propose a stochastic differential equation model in order to consider the OSS component collision of open source solution. Moreover, we assume that the software fault-detection rate depends on the time, and the software fault-report phenomena on the binding phase of open source solution keep an irregular state. Especially, we derive several performability assessment measures from our stochastic differential equation model. Also, we show that the performability analysis proposed here can assist improvement of quality for open source solution developed under several OSS’s.

2. Model Description

Let \( N(t) \) be the number of detected faults in the open source solution by binding time \( t (t \geq 0) \). Suppose that \( N(t) \) takes on continuous real values. Since latent faults in the open source solution are detected and eliminated during the binding phase, \( N(t) \) gradually increases as the testing procedures go on. Thus, under common assumptions for software reliability growth modeling, it is considered the conventional linear differential equation.

Generally, it is difficult for users to use all functions in open source solution, because the connection state among open source components is unstable in the binding-phase of open source solution. Considering the characteristic of open source solution, the software fault-report phenomena keep an irregular state in the early stage of binding-phase. Moreover, the addition and deletion of software components are repeated under the development of each OSS system, i.e., we consider that the software failure intensity depends on the time. Therefore, we consider the following stochastic differential equation[7, 8]:

\[
\frac{dN(t)}{dt} = \left\{ b(t) + \sigma \gamma(t) \mu(t) \right\} \{ a - N(t) \},
\]

where \( b(t) \) is the software fault-detection rate at binding time \( t \) and a non-negative function, \( a \) the expected number of the initial inherent faults, \( \sigma \) is a positive constant representing a magnitude of the irregular fluctuation, \( \gamma(t) \) a standardized Gaussian white noise, and \( \mu(t) \) the collision level function of OSS component.

By using Ito’s formula, the solution of Eq.(1) is obtained under the initial condition
Performability Analysis Considering Debugging Behaviors for Open Source Solution

\[ N(0) = 0 \] as follows[9],[10]:

\[ N(t) = a \left[ 1 - \exp \left\{ - \int_0^t b(s) ds - \sigma \mu(t) W(t) \right\} \right], \quad (2) \]

where \( W(t) \) is a one-dimensional Wiener process which is formally defined as an integration of the white noise \( \gamma(t) \) with respect to time \( t \).

Using solution process \( N(t) \) in Eq.(2), several software reliability measures can be derived.

Moreover, the intensity of inherent software failures in case of \( \beta(t) \equiv \beta_1(t) \) and \( \beta(t) \equiv \beta_2(t) \) are respectively defined as:

\[
\int_0^t \beta_1(t) ds = \frac{dN_1(t)}{dt} \equiv \frac{dH_1(t)}{dt}, \quad (3)
\]

\[
\int_0^t \beta_2(t) ds = \frac{dN_2(t)}{dt} \equiv \frac{dH_2(t)}{dt}, \quad (4)
\]

\[ \mu(t) = e^{-\omega t}, \]  \( (5) \)

where \( H_1(t) \) and \( H_2(t) \) mean the exponential SRGM and the delayed S-shaped SRGM, respectively, based on nonhomogeneous Poisson process (NHPP), \( b \) is the software failure rate per inherent fault, and \( \omega \) the parameter of stability for the open source solution. Our modeling technique is based on the stochastic differential equations[9],[10]. We propose new approaches to the characteristics of open source solution, i.e., the binding-phase, the complex and large scale system, and the confusion period in open source solution. Our model can easily comprehend the characteristics of such as large scale open source solution by using Eq. (5).

Therefore, the transition probability distribution of these two models are obtained as follows:

\[ N_1(t) = a \left[ 1 - \exp \left\{ -bt - \sigma \mu(t) W(t) \right\} \right], \quad (6) \]

\[ N_2(t) = a \left[ 1 - (1+bt) \exp \left\{ -bt - \sigma \mu(t) W(t) \right\} \right], \quad (7) \]

3. Parameter Estimation

In this section, the estimation method of unknown parameters \( a, b, \) and \( \sigma \) in Eq. (2) is presented. The joint probability distribution function of the process \( N(t) \) is denoted as

\[ P(t_1, y_1; t_2, y_2; \cdots; t_K, y_K) \equiv \Pr \{ N(t_i) \leq y_i, \cdots, N(t_K) \leq y_K \mid N(t_0) = 0 \}. \quad (8) \]

Its density of Eq. (8) is denote as
\[ p(t_1, y_1; t_2, y_2; \cdots; t_K, y_K) = \frac{\partial^k P(t_1, y_1; t_2, y_2; \cdots; t_K, y_K)}{\partial y_1 \partial y_2 \cdots \partial y_K}. \]  

Since \( N(t) \) takes on continuous values, the likelihood function, \( l \), for the observed data \((t_k, y_k) (k = 1, 2, \cdots, K)\) is constructed as follows:

\[ l = p(t_1, y_1; t_2, y_2; \cdots; t_K, y_K). \]  

For convenience in mathematical manipulations, the following logarithmic likelihood function is used:

\[ L = \log l. \]  

The maximum-likelihood estimates \( \hat{a} \), \( \hat{b} \), and \( \hat{\sigma} \) are the values making \( L \) in Eq. (11) maximize. These can be obtained as the solutions of the following simultaneous likelihood equations:

\[ \begin{align*}
\frac{\partial L}{\partial a} &= \frac{\partial L}{\partial b} = \frac{\partial L}{\partial \sigma} = 0.
\end{align*} \]  

4. Software Reliability Assessment Measures

Considering the expected number of faults detected up to operation time \( t \), the density function of \( W(t) \) is given by:

\[ f(W(t)) = \frac{1}{\sqrt{2\pi}} \exp \left\{ -\frac{W(t)^2}{2t} \right\}. \]  

Information on the cumulative number of detected faults in the system is important to estimate the situation of the progress on the debugging procedures. Since \( N(t) \) is a random variable in the proposed model, its expected value can be a useful measure. It can be calculated from Eq.(2) as follows[9],[10]:

\[ E[N(t)] = a \left[ 1 - \exp \left\{ -\int_0^t s b(s)ds + \frac{\sigma^2 \mu(t)^2}{2} \right\} \right], \]  

where \( E[N(t)] \) is the expected number of faults detected up to time \( t \).

Also, the expected number of remaining faults can be obtained as follows:

\[ E[N_r(t)] = a - E[N(t)]. \]  

Moreover, the mean time between software failures is useful to measure the property of the frequency of software failure-occurrence (or fault-detection). Then, the instantaneous MTBF (denoted by \( \text{MTBF}_I \)) is approximately given by:

\[ \text{MTBF}_I(t) = \frac{1}{E \left[ \frac{dN(t)}{dt} \right]}. \]
5. Performability Assessment Measures for Open Source Solutions

It is important for the project managers to understand the performability in binding-phase for open source solution. In this paper, the performability ratio is given by the following equation as the performability measure for open source solution:

\[
PR(t) = \frac{t - MTTR(t) \cdot E[N(t)]}{t},
\]

where \(MTTR(t)\) the down time of open source solution at time \(t\). Generally, \(MTTR\) means the mean time to correction as follows:

\[
MTTR(t) = \int_{0}^{\infty} t \cdot \gamma(t) dt.
\]

In this paper, we define the density function types of correction time as follows:

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

where \(\lambda\) is the rate parameter of the exponential distribution. Similarly, the sample path of performability ratio for open source solution is given as follows:

\[
PR_{\text{noise}}(t) = \frac{\lambda t - N(t)}{\lambda t}.
\]

6. Numerical Examples

We focus on a large scale open source solution based on the Apache HTTP Server [11], Apache Tomcat [12], MySQL [13] and JSP (JavaServer Pages). The fault-count data used in this paper are collected in the bug tracking system on the website of each open source project.

6.1 Reliability Assessment Results

The sample paths of the estimated numbers of detected faults in Eq. (2), \(\hat{N}(t)\)'s, in case of \(b(t) \equiv \hat{b}_1(t)\) and \(b(t) \equiv \hat{b}_2(t)\) are shown in Figs. 1 and 2, approximately. From Figs. 1 and 2, we can confirm that the sample path has variable in the early binding-phase as the interesting aspect of large-scale open source solution, i.e., our model can cover the collision status in the binding-phase of OSS's.
6.2 Performability Assessment Results

The performability ratio, $PR(t)$'s, in case of $b(t) \equiv b_1(t)$ and $b(t) \equiv b_2(t)$ are shown in Figs. 3 and 4, approximately. Moreover, Figs. 5 and 6 show the sample path of the performability ratio, $PR_{noise}(t)$'s, in case of $b(t) \equiv b_1(t)$ and $b(t) \equiv b_2(t)$. From Figs. 3–6, we found that the performability of open source solution growth as the binding procedures go on. Especially, Figs. 5 and 6 show that the performability ratio's decrease as the influence of the inflection point of S-shaped curve in the early binding phase.
Figure 3: The Estimated Performability Ratio, $PR(t)$, in case of $b(t) \equiv b_1(t)$.

Figure 4: The Sample Path of Performability Ratio, $PR_{noise}(t)$, in case of $b(t) \equiv b_1(t)$.

Figure 5: The Estimated Performability Ratio, $PR(t)$, in case of $b(t) \equiv b_2(t)$.
Figure 6: The Sample Path of Performability Ratio, $PR_{\text{noise}}(t)$, in case of $b(t) \equiv b_2(t)$.

7. Conclusions

We have focused on the open source solution which is known as the large scale software system, and discussed the method of performability assessment for the open source solution developed under several OSS’s.

At present, a new paradigm of distributed development typified by such open source projects will evolve at a rapid pace in the future. Especially, it is difficult for the software managers to assess the performability for the large scale open source solution as a typical case of next-generation distributed development paradigm. Our method may be useful as the method of performability assessment for the large scale open source solution.

Acknowledgements: This work was supported in part by the Grant-in-Aid for Scientific Research (C), Grant No. 24500066 and No. 22510150 from the Ministry of Education, Culture, Sports, Science, and Technology of Japan.

References

Performability Analysis Considering Debugging Behaviors for Open Source Solution


Yoshinobu Tamura received his 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, and 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.

Shigeru Yamada received his B.S.E., M.S., and Ph.D. degrees from Hiroshima University in 1975, 1977, and 1985, respectively. From 1988 to 1993, he was an associate professor at the Faculty of Engineering of Hiroshima University, Japan. Since 1993, he is a professor at the Department of Social Management Engineering, Graduate School of Engineering, Tottori University, Tottori-shi, Japan. He has published numerous technical papers in the area of software reliability models, project management, reliability engineering, and quality control. He has authored several books that include: 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), and Elements of Software Reliability (Kyoritsu Shuppan, 2011). 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 Paper Award from the Reliability Engineering Association of Japan in 1999, the International Leadership Award in Reliability Engg. 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 International Leadership and Pioneering Research Award in Software Reliability Engineering from the SREQOM/ICQRIT in 2009, and the Best Paper Award from the IEEE Reliability Society Japan Chapter in 2012. He is a regular member of the IPSJ, the ORSJ, the Japan SIAM, the REAJ, JIMA, the JSQC, the SPM Japan, and the IEEE.