American Journal of Operations Research, 2011, 1, 25-32
doi:10.4236/ajor.2011.12004 Published Online June 2011 (http://www.SciRP.org/journal/ajor/)

Component-Oriented Reliability Analysis Based on Hierarchical Bayesian Model for an Open Source Software

Yoshinobu Tamura[1], Hidemitsu Takehara[2], Shigeru Yamada[2]
[1]*Graduate School of Science and Engineering, Yamaguchi University, Ube, Japan*
[2]*Graduate School of Engineering, Tottori University, Tottori, Japan*
E-mail: *tamura@yamaguchi-u.ac.jp, M09T7016M@edu.tottori-u.ac.jp, yamada@sse.tottori-u.ac.jp*
Received March 11, 2011; *revised April* 10, 2011; *accepted May* 9, 2011

Abstract

The successful experience of adopting distributed development models in such open source projects includes GNU/Linux operating system, Apache HTTP server, Android, BusyBox, and so on. 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. We propose a method of component-oriented reliability assessment based on hierarchical Bayesian model and Markov chain Monte Carlo methods. Especially, we focus on the fault-detection rate for each component reported to the bug tracking system. We can assess the reliability for the whole open source software system by using the confidence interval for each component. Also, we analyze actual software fault-count data to show numerical examples of reliability assessment for OSS.

Keywords: Open Source Software, Reliability, Bayesian Model, Markov Chain Monte Carlo Method

1. Introduction

Software development environment has changed into new development paradigms such as concurrent distributed development environment and the so-called open source project by using network computing technologies [1]. Especially, such OSS (Open Source Software) systems which serve as key components of critical infrastructures in the society are still ever-expanding now [2]. The methodology of the object-oriented design and analysis is a feature of distributed development environment and greatly successful in the field of programming-language, simulation, GUI (graphical user interface), and constructing on database in the software development. A general idea of the object-oriented design and analysis is developed as a technique which can easily construct and maintain the complex system. The successful experience of adopting the distributed development model in such open source projects includes GNU/Linux operating system [2]. However, the poor handling of the quality and customer support prohibit the progress of OSS. We focus on the problems in the software quality, which prohibit the progress of OSS.

Especially, many software reliability growth models (SRGM's) [3] 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 testing management for a new distributed development paradigm as typified by the open source project has only a few presented [4-8]. 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 connection status of each component, degree of maturation of OSS, and so on [9,10].

Especially, OSS is composed of several software components as a feature of distributed development environment. In such cases, it is appropriate to apply the method of component based reliability assessment rather than one of reliability assessment based on SRGM's. Many SRGM's are assumed to be suitable for the system testing phase of software development. On the other hand, it is difficult to apply SRGM's to OSS, because OSS development style has not the typical software development environment, *i.e.*, OSS development cycle has no testing phase. Moreover, OSS is developed under a combination of many software components. Therefore, it is important for software developers to confirm the static state of each component in OSS development phase from the standpoint of reliability assessment [6-8]. The char-

acteristics in terms of the reliability assessment for OSS's are shown as follows [11,12]:

- OSS development cycle has no testing phase;
- The cumulative number of detected faults can not converge to a finite value;
- It is difficult to apply SRGM's to the development cycle of OSS;
- OSS is developed under a combination of many software components;
- Many software components of OSS are developed by the geographically-dispersed software developers;
- OSS's are grouped into several categories, *i.e.*, application software such as Firefox Web browser, server software such as Apache HTTP server, embedded system software such as Android, operating system software such as Linux.

In this paper, we focus on an OSS developed under open source project. We discuss the method of component-oriented software reliability assessment considering the fault-detection rate of each component based on Bayesian theory and Markov chain Monte Carlo methods (MCMC). It is important to understand the static state of OSS, *i.e.*, the connection status of each component. We consider the method of reliability assessment for the whole OSS system by using the data of proportion for fault-detection rate in terms of the software components. Especially, we estimate the predicted distributions by using MCMC. Then, we use the data of proportion data for fault-detection rate in terms of the software components on the bug tracking system as the sample data. Also, we analyze actual software fault-count data to show numerical examples of software reliability assessment for the OSS. Especially, we derive the confidence interval for each component. Then, we show that the proposed method can assist improvement of quality for OSS. Our method may be useful for the software testing manager to assess the static state of the whole OSS system automatically.

2. Estimation of Predicted Distribution Based on Bayesian Theory

We apply a Bayesian theory to the data in terms of fault-detection rate of each OSS components. Let y_t be the proportion data of the fault-detection rate in the OSS by operational time t. Also, θ_t is the parameter of the specific distribution at operational time t. We estimate θ_t by using y_t. In this case, we use the prior distribution up to time $(t-1)$. As an example, the updated data is given by the following equation based on Bayesian theory in case that we have knowledge of the prior information θ independently of data D.

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{\int p(D \mid \theta)p(\theta)d\theta} \propto p(D \mid \theta)p(\theta). \tag{1}$$

In this paper, we estimate θ_t by using the data of proportion for past fault detection rate in order to estimate θ_t for the sequential data y_t. Then, we can derive the following equation from Equation (1):

$$\begin{aligned} &p(\theta_t \mid y_1, y_2, \cdots, y_t) \\ &= p(y_t \mid \theta_t)p(\theta_t \mid y_1, y_2, \cdots, y_{t-1}) \\ &\Big/ \int p(y_t \mid \theta_t)p(\theta_t \mid y_1, y_2, \cdots, y_{t-1})\mathrm{d}\theta_t . \end{aligned} \tag{2}$$

According to Equation (2), $p(\theta_t \mid y_1, y_2, \cdots, y_t)$ is updated on a real-time basis from $p(\theta_{t-1} \mid y_1, y_2, \cdots, y_{t-1})$. Therefore, we define as follows:

$$\begin{aligned} &p(\theta_t \mid y_1, y_2, \cdots, y_t) \\ &= p(y_t \mid \theta_t)\int p(\theta_t \mid \theta_{t-1})p(\theta_{t-1} \mid y_1, y_2, \cdots, y_{t-1})\mathrm{d}\theta_{t-1} \\ &\Big/ \int p(y_t \mid \theta_t)p(\theta_t \mid y_1, y_2, \cdots, y_{t-1})\mathrm{d}\theta_t . \end{aligned} \tag{3}$$

Equation (3) means the probability at operational time t obtained from θ_{t-1} at operational time $(t-1)$.

Then, we assume the simple case as follows:

$$\theta_t = \theta_{t-1} + \varepsilon_{t-1}, \tag{4}$$

where ε_t is the independent Gaussian noise at operational time t [13].

3. Hierarchical Bayesian Model

In this paper, we assume the data trend of proportion for the fault-detection rate as the following probability density function of normal distribution for simplicity:

$$f(x) = \frac{1}{\sqrt{2\pi t}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right), \tag{5}$$

where μ is the mean value and σ the standard deviation. We consider the hierarchical Bayesian model based on the prior distribution and hyper prior distribution composed of μ and σ.

Then, we can obtain the following equation from Equation (1).

$$\begin{aligned} &p(\mu, \sigma \mid D) \\ &= p(D \mid \mu, \sigma)p(\mu \mid \sigma)p(\sigma) \\ &\Big/ \iint p(D \mid \mu, \sigma)p(\mu \mid \sigma)p(\sigma)\mathrm{d}\mu\mathrm{d}\sigma \\ &\propto p(D \mid \mu, \sigma)p(\mu \mid \sigma)p(\sigma). \end{aligned} \tag{6}$$

Therefore, we can derive as follows:

$$\begin{aligned} &p(\mu_t, \sigma_t \mid y_1, y_2, \cdots, y_t) \\ &= \big(p(y_t \mid \mu_t, \sigma_t)p(\mu_t \mid \sigma_t) \\ &\cdot p(\sigma_t \mid y_1, y_2, \cdots, y_{t-1}) \big) \\ &\Big/ \Big(\iint p(y_t \mid \mu_t, \sigma_t)p(\mu_t \mid \sigma_t) \\ &p(\sigma_t \mid y_1, y_2, \cdots, y_{t-1})\mathrm{d}\mu_t\mathrm{d}\sigma_t \Big). \end{aligned} \tag{7}$$

According to Equation (7), $p(\mu_t, \sigma_t \mid y_1, y_2, \cdots, y_t)$ is updated on a real-time basis from $p(\mu_{t-1}, \sigma_{t-1} \mid y_1, y_2, \cdots, y_{t-1})$. Therefore, we can obtain as follows:

$$\begin{aligned} & p(\mu_t, \sigma_t \mid y_1, y_2, \cdots, y_t) \\ & = \big(p(y_t \mid \mu_t, \sigma_t) p(\mu_t \mid \sigma_t) \\ & \cdot \int p(\sigma_t \mid \sigma_{t-1}) p(\sigma_t \mid y_1, y_2, \cdots, y_{t-1}) \mathrm{d}\sigma_{t-1} \big) \\ & \Big/ \Big(\iint p(y_t \mid \mu_t, \sigma_t) p(\mu_t \mid \sigma_t) \\ & \cdot p(\sigma_t \mid y_1, y_2, \cdots, y_{t-1}) \mathrm{d}\mu_t \mathrm{d}\sigma_t \Big). \end{aligned} \tag{8}$$

Equation (8) means the probability at operational time *t* obtained from the operational time $(t-1)$. We can estimate μ and σ at operational time *t* from Equation (8).

Also, we can derive the confidence interval from the estimated mean value $\hat{\mu}$ and standard deviation $\hat{\sigma}$. In case of the upper side probability $100\alpha\%$ and the degree of freedom $(n-1)$, we can obtain the upper and lower confidence limits for the estimated confidence interval as follows:

$$\hat{\mu} \pm t_{n-1}(1-\alpha) \frac{\hat{\sigma}}{\sqrt{n}}, \tag{9}$$

where $t_k(\alpha)$ is the value of t distribution in case of the confidence interval $(1-\alpha)$ at the degree of freedom k. Also, n means the total number of data.

4. MCMC

It is one of the sampling method of the probability distribution based on Markov chain by the random number generation. Basically, it is difficult to take a sample of random variable from the multivariate distribution. However, we can easily take the probability sample from the objective probability by using MCMC [14,15]. Several MCMC algorithms have been proposed by several researchers in order to solve these problems, *i.e.*, Metropolis-Hastings (MH) and Gibbs Sampler. Gibbs Sampler is the extended method of MH algorithm. We apply the Metropolis-Hastings (MH) algorithm in this paper, because MH algorithm has simple structure, and widely used in many research fields.

The flow of MH algorithm is shown in **Figure 1**. Also, the procedures of MH algorithm is as follows:

- Generate θ' by using the applied density $p(\theta, \theta')$ in case of $\theta_t = \theta$.
- In case of $\alpha(\theta, \theta') > u$, replace θ' by θ_{t+1}. In case of $\alpha(\theta, \theta') \le u$, apply $\theta_{t+1} = \theta$.
- Continue the above mentioned process without the initial value dependence.

In this paper, we assume that $p(\theta_t)$ is $p(\mu_t, \sigma_t)$. Also, θ' means μ' and σ'. Therefore, $\alpha(\theta, \theta')$ is given by the following equation in this paper:

$$\alpha = \frac{p(\mu', \sigma' \mid y_1, y_2, \cdots, y_t)}{p(\mu_t, \sigma_t \mid y_1, y_2, \cdots, y_t)}. \tag{10}$$

5. Numerical Examples

There are many open source projects around the world. In particular, we focus on an large scale open source solution based on the Apache HTTP Server [16]. The fault-count data used in this paper are collected in the bug tracking system on the website of each open source project.

5.1. The Estimation Results Based on MCMC

The data of proportion for actual fault-detection rate for each component in Apache HTTP Server is shown in **Table 1**. We use the data from January 2008 to September 2010. **Table 1** shows the data of proportion for actual fault-detection rate for each month. Also, we apply "Core", "Documentation", "mod_ssl", "mod_proxy", and "Build" as the major components. We focus on the data of all platform for Apache HTTP Server 2 version. We assume that a unit of time is week, because these results and computational times show little change if the unit of time is day in terms of the software fault data sets.

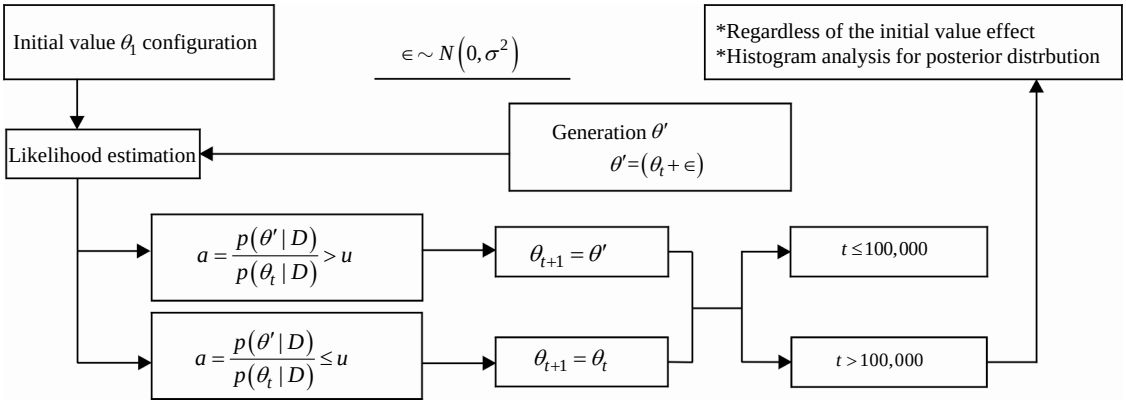


Figure 1. The flow diagram of MH algorithm.

Table 1. The data of proportion for actual fault-detection rate for each component in Apache HTTP Server.

Date	Core	Documentation	mod_ssl	mod_proxy	Build
Jan-08	0.500 00	0.050 00	0.150 00	0.250 00	0.050 00
Feb-08	0.357 14	0.142 86	0.071 43	0.214 29	0.214 29
Mar-08	0.153 85	0.307 69	0.076 92	0.230 77	0.230 77
Apr-08	0.269 23	0.307 69	0.153 85	0.192 31	0.076 92
May-08	0.285 71	0.238 10	0.285 71	0.095 24	0.095.24
Jun-08	0.375 00	0.187 50	0.000 00	0.062 50	0.375.00
Jul-08	0.280 00	0.120 00	0.080 00	0.320 00	0.200.00
Aug-08	0.230 77	0.076 92	0.307 69	0.307 69	0.076 92
Sep-08	0.277 78	0.166 67	0.222 22	0.111 11	0.222 22
Oct-08	0.380 95	0.142 86	0.142 86	0.047 62	0.285 71
Nov-08	0.214 29	0.214 29	0.071 43	0.214 29	0.285 71
Dec-08	0.176 47	0.352 94	0.294 12	0.117 65	0.058 82
Jan-09	0.315 79	0.421 05	0.105 26	0.052 63	0.105 26
Feb-09	0.200 00	0.133 33	0.133 33	0.333 33	0.200 00
Mar-09	0.285 71	0.285 71	0.142 86	0.142 86	0.142 86
Apr-09	0.500 00	0.000 00	0.250 00	0.250 00	0.000 00
May-09	0.250 00	0.250 00	0.250 00	0.166 67	0.08333
Jun-09	0.235 29	0.352 94	0.294 12	0.117 65	0.000 00
Jul-09	0.200 00	0.300 00	0.200 00	0.100 00	0.20000
Aug-09	0.285 71	0.357 14	0.071 43	0.142 86	0.14286
Sep-09	0.625 00	0.12500	0.125 00	0.125 00	0.00000
Oct-09	0.222 22	0.55556	0.11111	0.000 00	0.111 11
Nov-09	0.200 00	0.06667	0.46667	0.200 00	0.066 67
Dec-09	0.437 50	0.18750	0.062 50	0.187 50	0.125 00
Jan-10	0.222 22	0.33333	0.222 22	0.111 11	0.111 11
Feb-10	0.363 64	0.18182	0.363 64	0.090 91	0.000 00
Mar-10	0.315 79	0.052 63	0.210 53	0.210 53	0.210 53
Apr-10	0.666 67	0.111 11	0.111 11	0.000 00	0.111 11
May-10	0.000 00	0.500 00	0.375 00	0.000 00	0.125 00
Jun-10	0.545 45	0.181 82	0.181 82	0.090 91	0.000 00
Jul-10	0.333 33	0.222 22	0.333 33	0.000 00	0.111 11
Aug-10	0.411 76	0.235 29	0.235 29	0.000 00	0.117 65
Sep-10	0.300 00	0.500 00	0.200 00	0.000 00	0.000 00

Table 2. Comparison of the estimate with the actual data.

	Mean		Standard Deviation	
	Estimate	Actual	Estimate	Actual
Core	0.3152	0.3157	0.1445	0.1386
Document	0.2325	0.2321	0.1440	0.1377
mod_ssl	0.1908	0.1910	0.1127	0.1078
mod_proxy	0.1357	0.1359	0.1020	0.0978
Build	0.1251	0. 1253	0.0981	0.0941

We show the estimation results based on MCMC for each component in **Figures 2-6**, respectively. Moreover, the comparison results of the estimates with the actual data are shown in **Table 2**. Above mentioned results, we can find that the level of fault-detection rate for "Core" component is largest. On the other hand, we can find that the level of fault detection rate for "Build" component is smallest. Therefore, we can confirm that "Core" component is the most affected one for the whole OSS system. Moreover, we can confirm that the standard deviation of fault importance level for "Document" is large. Thereby, there is variation in the data of proportion for actual fault-detection rate. Also, the estimation results of **Table 2** is shown to be optimistic results in terms of the standard deviations.

We show the estimation results based on MCMC for each component in **Figures 2-6**, respectively. Above the mentioned results, we can find that the level of fault detection rate for "Core" component is largest. On the other hand, we can find that the level of fault detection rate for "Build" component is smallest. Therefore, we can confirm that "Core" component is the most affected one for the whole OSS system. Moreover, we can confirm that the standard deviation of fault importance level for "Document" is large. Thereby, there is variation in the data of proportion for actual fault-detection rate.

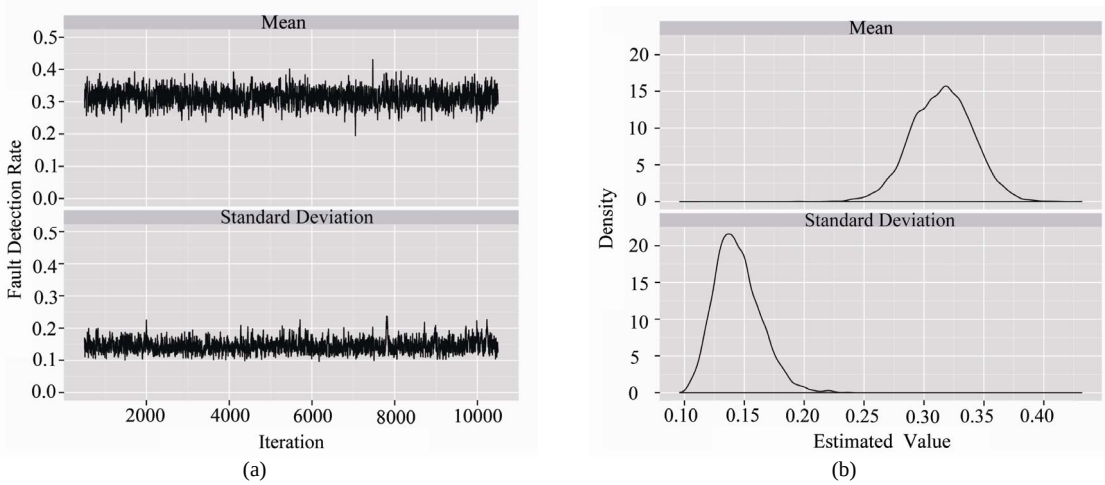
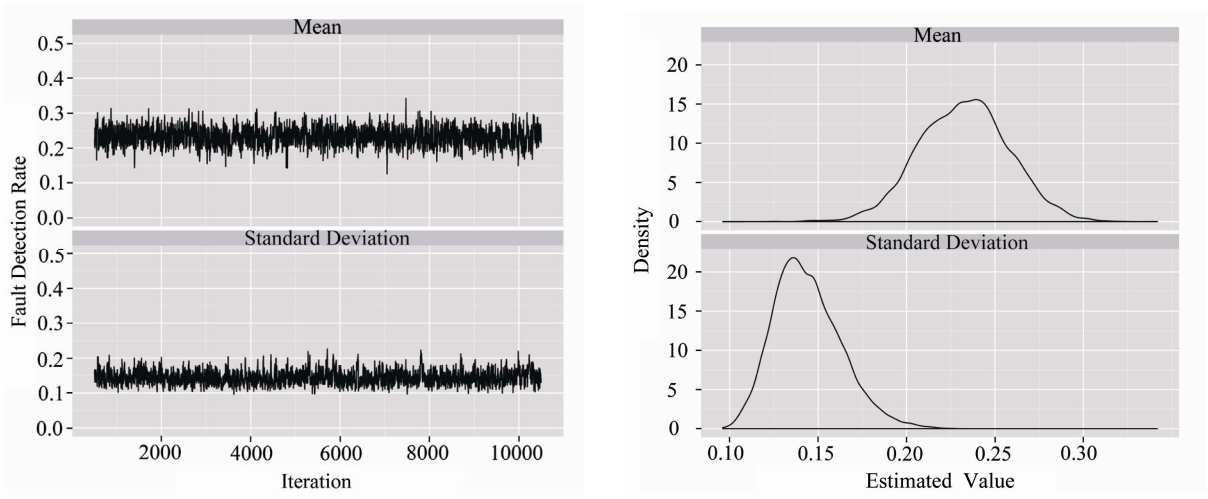
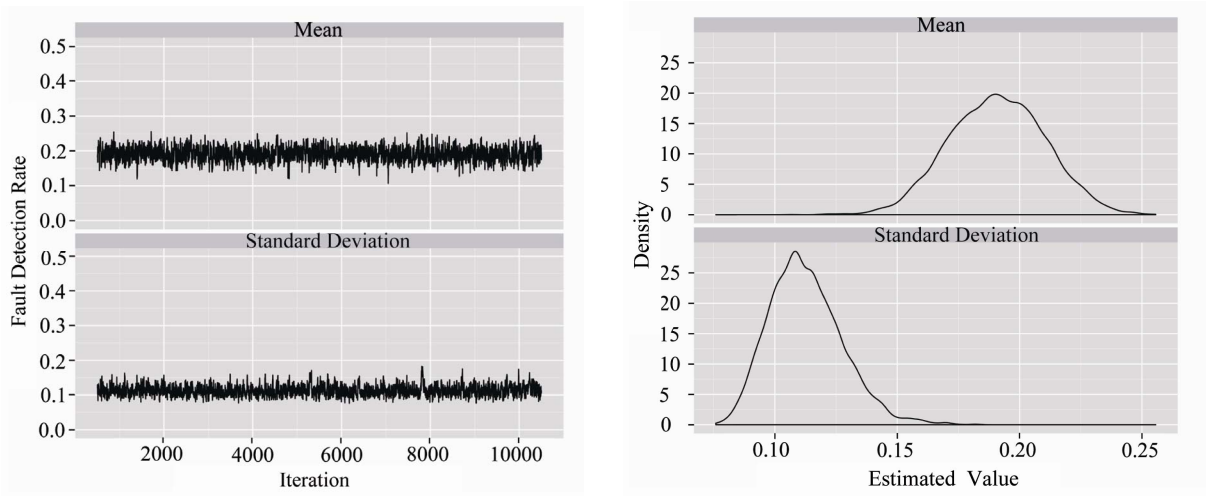


Figure 2. The estimation results for Core component.



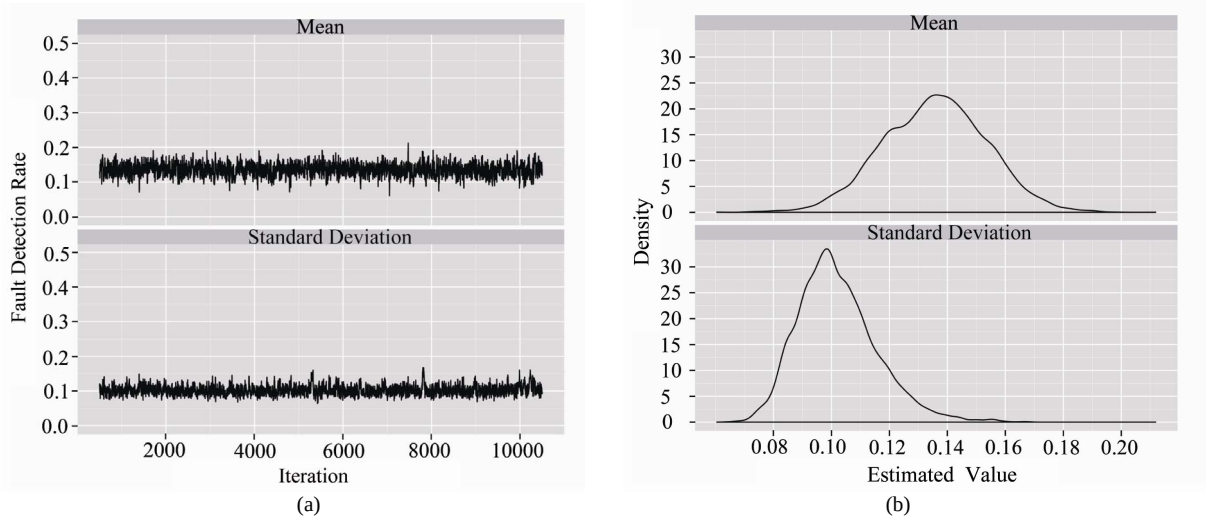
(a) (b)

Figure 3. The estimation results for Documentation component.



(a) (b)

Figure 4. The estimation results for mod_ssl component.



(a) (b)

Figure 5. The estimation results for mod_proxy component.

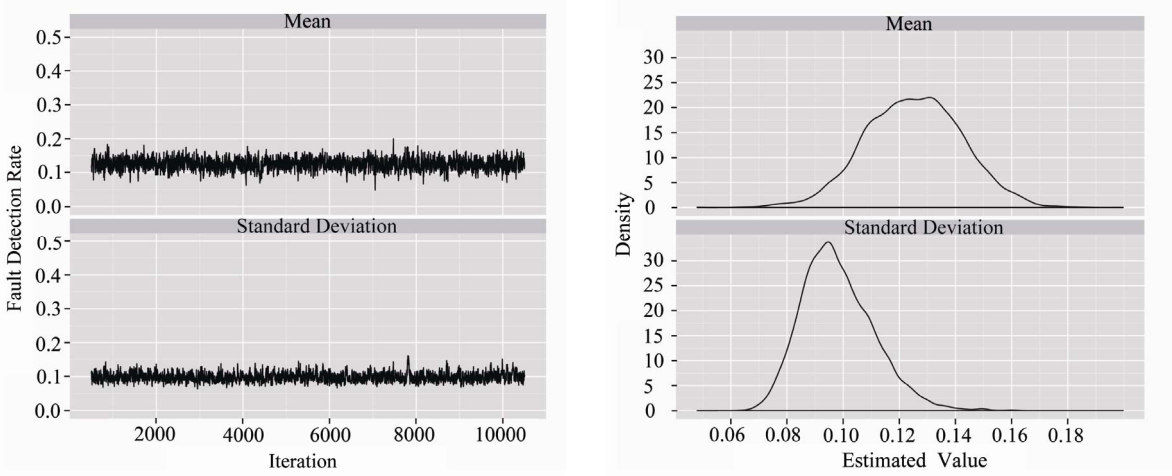


Figure 6. The estimation results for Build component.

5.2. The Estimation Results Based on MCMC with Time Variation Considering Confidence Interval

In this section, we consider the case of $n=24, \alpha=0.95$. Then, 95% confidence interval is given by the following equation:

$$\hat{\mu} \pm t_{23}(0.05)\frac{\hat{\sigma}}{24}. \tag{7}$$

The estimation results based on MCMC with time variation of the data of proportion for each component are shown in **Figures 7-11**, respectively. From **Figures 7-11**, we can confirm that "Core" component is constant in small width of the confidence interval. Also, "Core" component remains in the large value continuously. These results mean that the open source project keeps a high active state. Therefore, we consider that the focused OSS system is stable in terms of the occurrence rate of "Core" component. On the other hand, "mod_ssl" and "mod_proxy" components decrease in width of the confidence interval, because the open source project is proceeding without problems according to be removed the faults of small components.

6. Concluding Remarks

In this paper, we have focused on the reliability of OSS. Moreover, we have proposed the method of component-oriented software reliability assessment based on the hierarchical Bayesian model and MCMC in order to estimate the predicted distributions for each component of OSS. Especially, we have assumed the data of proportion for the fault-detection rate as the probability density function of normal distribution. Also, we have analyzed actual software fault-count data to show numerical examples of component-oriented software reliability assessment for OSS.

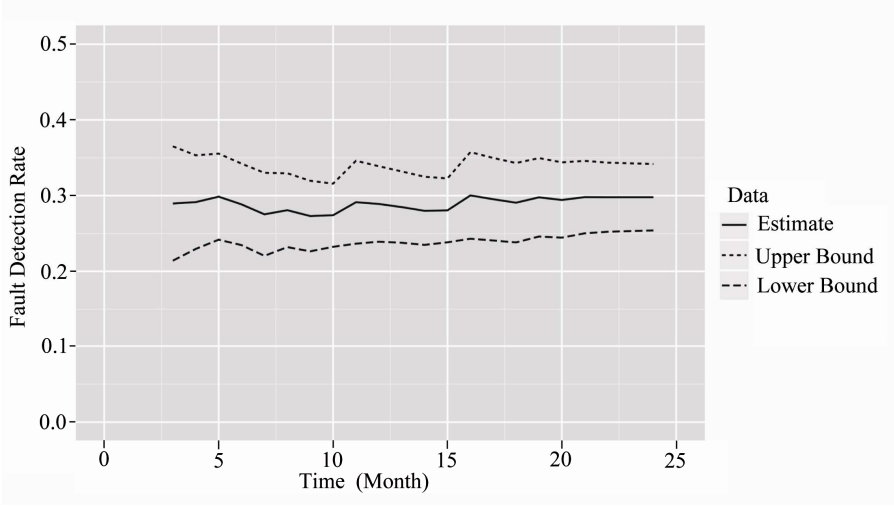


Figure 7. The estimation results of confidence interval for Core component.

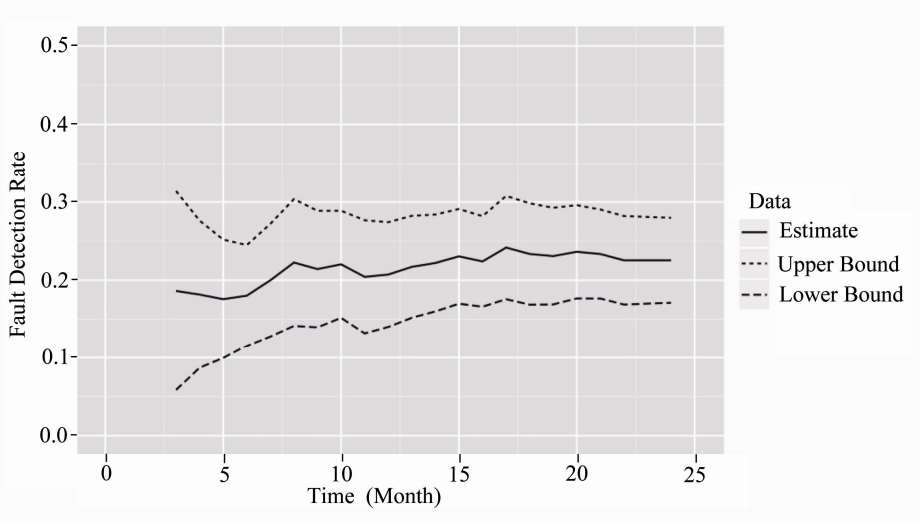


Figure 8. The estimation results of confidence interval for Documentation component.

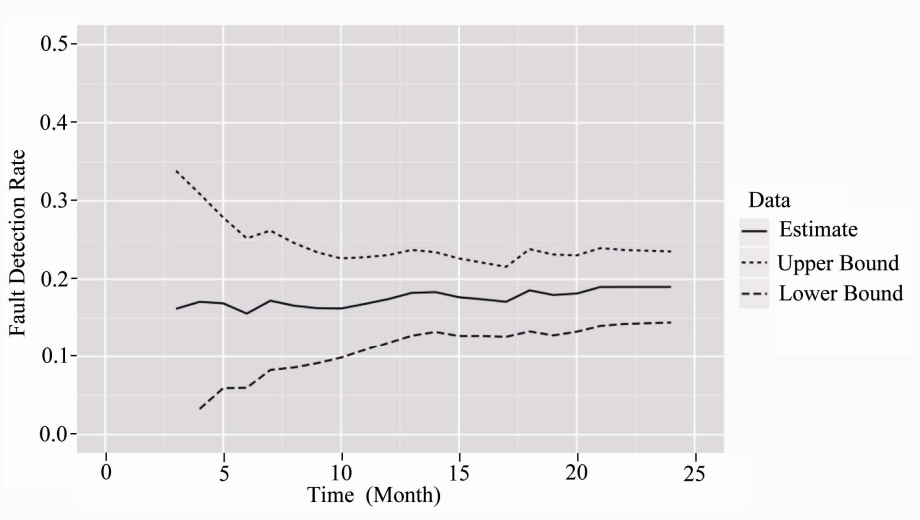


Figure 9. The estimation results of confidence interval for mod_ssl component.

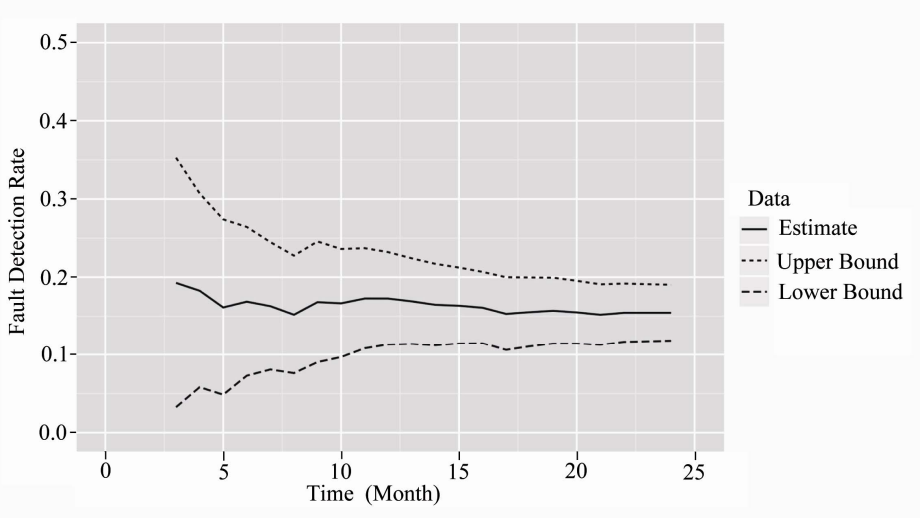


Figure 10. The estimation results of confidence interval for mod_proxy component.

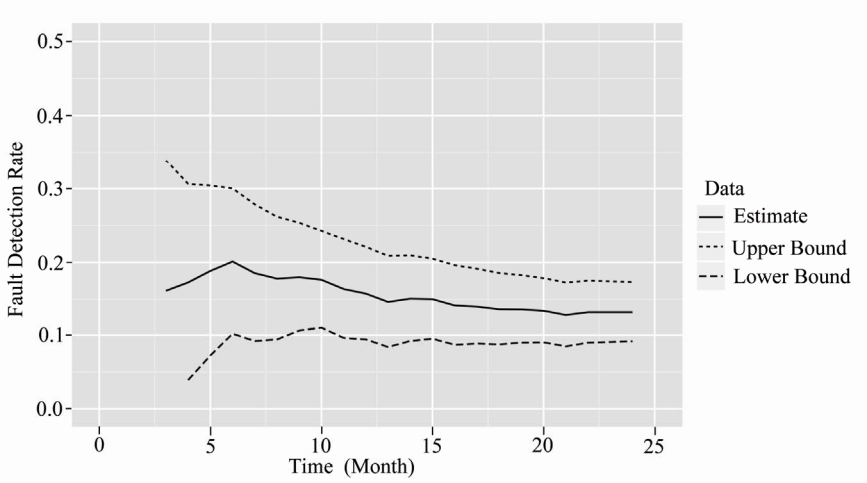


Figure 11. The estimation results of confidence interval for Build component.

Finally, we have focused on fault-detection rate for fault importance level of OSS. By using our method, the software testing manager can assess the static state of OSS. Our method may be useful as the method of component-oriented reliability assessment for OSS.

7. Acknowledgements

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

8. References

[1] L. T. Vaughn, "Client/Server System Design and Implementation," McGraw-Hill, New York, 1994.

[2] E-Soft Inc., "Internet Research Reports," 2010. http://www.securityspace.com/sspace/

[3] S. Yamada, "Elements of Software Reliability-Modeling Approach (in Japanese)," Kyoritsu-Shuppan, Tokyo, 2011.

[4] A. D. MacCormack, J. Rusnak and C. Y. Baldwin, "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," *Informs Journal of Management Science*, Vol. 52, No. 7, 2006, pp. 1015-1030.

[5] G. Kuk, "Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List," *Informs Journal of Management Science*, Vol. 52, No. 7, 2006, pp. 1031-1042.

[6] Y. Zhoum and J. Davis, "Open Source Software Reliability Model: An Empirical Approach," *Proceedings of the Workshop on Open Source Software Engineering*, Vol. 30, No. 4, 2005, pp. 67-72,

[7] P. Li, M. Shaw, J. Herbsleb, B. Ray and P. Santhanam, "Empirical Evaluation of Defect Projection Models for Widely-Deployed Production Software Systems," *Proceedings of the* 12*th International Symposium on the Foundations of Software Engineering*, New York, November 2004, pp. 263-272.

[8] J. Norris, "Mission-Critical Development with Open Source Software," *IEEE Software Magazine*, Vol. 21, No. 1, 2004, pp. 42-49.

[9] Y. Tamura and S. Yamada, "Software Reliability Assessment and Optimal Version-Upgrade Problem for Open Source Software," *Proceedings of the* 2007 *IEEE International Conference on Systems, Man, and Cybernetics*, Montreal, 7-10 October 2007, pp. 1333-1338. doi:10.1109/ICSMC.2007.4413582

[10] Y. Tamura and S. Yamada, "A Method of User-Oriented Reliability Assessment for Open Source Software and its Applications," *Proceedings of the* 2006 *IEEE International Conference on Systems, Man, and Cybernetics*, Taipei, 8-11 October 2006, pp. 2185-2190. doi:10.1109/ICSMC.2006.385185

[11] D. Bosio, B. Littlewood, L. Strigini and M. J. Newby, "Advantages of Open Source Processes for Reliability: Clarifying the Issues," *Proceedings of the Open Source Software Development Workshop*, Newcastle, 25-26 February 2002, pp. 30-46.

[12] F. Zou and J. Davis, "Analyzing and Modeling Open Source Software Bug Report Data," *Proceedings of the* 19*th Australian Conference on Software Engineering*, Washington, D.C., 26-28 March 2008, pp. 461-469.

[13] T. Matsumoto, "Implementations of Bayesian Learning (in Japanese)," *Journal of the Institute of Electronics, Information and Communication Engineers*, Vol. 92, No. 10, 2009, pp. 853-860.

[14] B. P. Carlin and S. Chib, "Bayesian Model Choice via Markov Chain Monte Carlo," *Journal of Royal Statistical Society*: *Series B* (*Methodological*), Vol. 57, No. 3, 1995, pp. 473-484.

[15] P. J. Green, "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination," *Journal of Biometrika*, Vol. 82, No. 4, 1995, pp. 711-732. doi:10.1093/biomet/82.4.711

[16] The Apache HTTP Server Project, "The Apache Software Foundation," 2010. http://httpd.apache.org/

