International Journal of Mathematical, Engineering and Management Sciences
Vol. 5, No. 4, 591-601, 2020
https://doi.org/10.33889/IJMEMS.2020.5.4.048

Stability Assessment Method Considering Fault Fixing Time in Open Source Project

Hironobu Sone
Graduate School of Integrative Science and Engineering,
Tokyo City University, Tokyo, Japan.
Corresponding author: g1881827@tcu.ac.jp

Yoshinobu Tamura
Department of Intelligent Systems,
Tokyo City University, Tokyo, Japan.
E-mail: tamuray@tcu.ac.jp

Shigeru Yamada
Graduate School of Engineering,
Tottori University, Tottori-shi, Japan.
E-mail: yamada@tottori-u.ac.jp

(Received November 21, 2019; Accepted March 12, 2020)

Abstract
Recently, open source software (OSS) are adopted various situations because of quick delivery, cost reduction and standardization of systems. Many OSS are developed under the peculiar development style known as bazaar method. According to this method, faults are detected and fixed by users and developers around the world, and the fixed result will be reflected in the next release. Also, the fix time of faults tends to be shorter as the development of OSS progresses. However, several large-scale open source projects have a problem that faults fixing takes a lot of time because faults corrector cannot handle many faults reports quickly. Furthermore, imperfect fault fixing sometimes occurs because the fault fixing is performed by various people and environments. Therefore, OSS users and project managers need to know the stability degree of open source projects by grasping the fault fixing time. In this paper, for assessment stability of large-scale open source project, we derive the imperfect fault fixing probability and the transition probability distribution. For derivation, we use the software reliability growth model based on the Wiener process considering that the fault fixing time in open source projects changes depending on various factors such as the fault reporting time and the assignees for fixing faults. In addition, we applied the proposed model to actual open source project data and examined the validity of the model.

Keywords- Reliability, Stochastic differential equation, Open source project.

1. Introduction
Source codes of open source software (OSS) are freely available for use, reuse, fix and re-distribution by the OSS users. Many OSS are known for their high performance and reliability although they are free of charge. Also, many IT companies often develop OSS for commercial use. In particular, OSS are developed by using the bazaar method (Raymond, 1999), the source code is implemented in public through the Internet. Then, OSS are promoted by an unspecified number of users and developers. The bug tracking system is also known as one of the systems used to develop OSS. Many fault information such as fix status, their details, and fix priorities are registered through the bug tracking system. Although OSS have been actively developed and used in recent years, there are problems in promoting open source projects. There are over 100 faults reported per day

in massive open source projects (Sun et al., 2010). Massive open source projects have a large number of fault reports. Then, it is difficult to quickly fix the faults (Hooimeijer et al., 2010). In addition, imperfect fault fixing sometimes occurs because the fault fixing is performed by various people and environments.

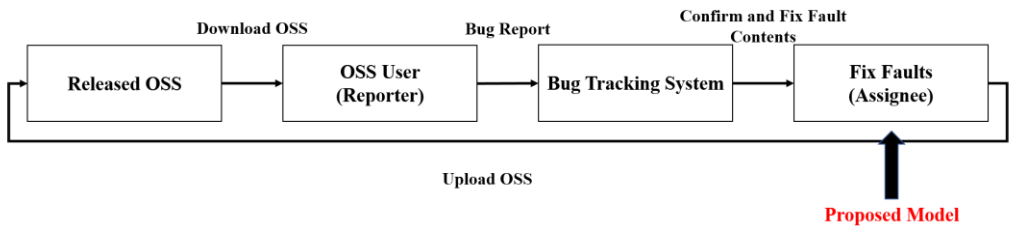


Figure 1. The OSS development using bug tracking system

In terms of OSS fixing speed, OSS users usually want to use OSS under the situation of stable development. There are many studies in the field of the stability assessment of open source projects.

Generally, as the development of OSS progresses, the number of unfixed faults and faults with long fixing time decreases. Then, OSS becomes stable. For stable OSS development, several previous studies predict the fix time of each reported fault (Bougie et al., 2010; Giger et al., 2010; Akbarinasaji et al., 2018). However, it is difficult to evaluate the stability of future open source projects by predicting the individual fault fixing time. By predicting the outlook for future projects, the OSS developers and managers can make a future plan. Based on the above, the purpose of this paper is to predict the required fault fixing time and evaluate the stability of open source projects based on this information.

In this paper, we derive the imperfect fault fixing probability and the transition probability distribution of the model used in this study in OSS development for assessing project stability. In particular, the imperfect fault fixing probability means probability of inefficient fault fixing. By deriving these results, we can obtain indicators to know when the project becomes stable. By grasping the stability of the project, the OSS manager can be linked to the stable operation of the project, and the OSS user can be linked to the decision making of the OSS selection. Especially, we focus on OSS development using bug tracking system in operational phase as depicted in Figure 1. In this paper, we aim to derive the imperfect fault fixing probability and the transition probability distribution using the exponential model and the delayed S-shaped model derived from the software reliability growth models (Musa et al., 1987; Lyu, 1996; Kapur et al., 2011; Yamada, 2014). The exponential model and the delayed S-shaped model are known as one of the famous software reliability growth models (Ranjan et al., 2019). By using the stochastic differential models, we can describe the irregulate situation such as OSS development. In particular, we derive the imperfect fault fixing probability and the transition probability distribution considering the characteristics of open source projects. Then, we assume that the number of developers and users change irregularly. We can easily discover the regularity from various factors in open source projects and apply mathematical models with multiple parameters. However, it is difficult to actually use these models in terms of parameter estimation. In this paper, we apply a stochastic differential equation model with noise based on the Wiener process considering the specific circumstances of open source projects. The proposed model will be able to evaluate the project quantitatively considering external factors from indirectly in open source projects.

2. Wiener Process Models for Assessment OSS Stability
Considering the characteristic of the method of fault fixing in open source projects, the time-dependent fault fixing effort expenditure phenomenon keeps an irregular state because OSS developers have different skills and developing environments respectively.

From above points, we discuss on the stochastic differential equation modeling for the open source projects. Then, let $\Omega(t)$ be the cumulative fault fixing time needed up to operational time $t(t \geq 0)$ from the start of the open source project. Suppose that $\Omega(t)$ takes on continuous real values. Since the estimated fault fixing time are observed during the operational phase of the open source project, $\Omega(t)$, gradually increases as the operational procedures go on. Based on software reliability growth modeling approach, the following linear differential equation in terms of fault fixing time can be formulated:

$$\frac{d\Omega(t)}{dt} = \beta(t)\{\alpha - \Omega(t)\}, \tag{1}$$

where $\beta(t)$ is the increase rate of fault fixing time at operational time t and a non-negative function, and α means the estimated fault fixing time required until the end of operation. Generally, $\Omega(t)$ means the number of faults in software. However, we consider $\Omega(t)$ as the cumulative fault fixing time for stability assessment of the software in terms of fixing time.

Therefore, we extend Eq. (1) to the following stochastic differential equation with Brownian motion (Wong, 1981):

$$\frac{d\Omega(t)}{dt} = \{\beta(t) + \sigma\nu(t)\}\{\alpha - \Omega(t)\}, \tag{2}$$

where σ is a positive constant representing a magnitude of the irregular fluctuation, and $\nu(t)$ a standardized Gaussian white noise. By using Itô's formula (Arnold, 1974), we can obtain the solution of Eq. (2) under the initial condition $\Omega(0) = 0$ as follows:

$$\Omega(t) = \alpha\left[1 - \exp\left\{-\int_0^t \beta(s)ds - \sigma\omega(t)\right\}\right], \tag{3}$$

where $\omega(t)$ is one-dimensional Wiener process which is formally defined as an integration of the white noise $\nu(t)$ with respect to time t. Moreover, we define the increase rate of fault fixing time in case of $\beta(t)$ defined as:

$$\int_0^t \beta(s)ds \doteq \frac{\frac{dF_*(t)}{dt}}{\alpha - F_*(t)}. \tag{4}$$

In this paper, we assume the following equations based on software reliability models $F_*(t)$ as the cumulative fault fixing time function of the proposed model:

$$F_e(t) = \alpha\left(1 - e^{-\beta t}\right), \tag{5}$$

$$F_s(t) = \alpha\{1 - (1 + \beta t)e^{-\beta t}\}. \tag{6}$$

where β is the increase rate of fault fixing time. Also, $\Omega_e(t)$ means the cumulative fault fixing time for the exponential software reliability growth model with $F_e(t)$. Similarly, $\Omega_s(t)$ is the cumulative fault fixing time for the delayed S-shaped software reliability growth model with $F_s(t)$.

Therefore, the cumulative fault fixing time up to time t are obtained as follows:

$$\Omega_e(t) = \alpha[1 - \exp\{-\beta t - \sigma\omega(t)\}], \tag{7}$$

$$\Omega_s(t) = \alpha[1 - (1 + \beta t)\exp\{-\beta t - \sigma\omega(t)\}]. \tag{8}$$

In this model, we assume that the parameter σ depends on several noises by external factors from several triggers in open source projects. Then, the expected cumulative fault fixing time spent up to time t are respectively obtained as follows:

$$\mathrm{E}[\Omega_e(t)] = \alpha\left[1 - \exp\left\{-\beta t + \frac{\sigma^2}{2}t\right\}\right], \tag{9}$$

$$\mathrm{E}[\Omega_s(t)] = \alpha\left[1 - (1 + \beta t)\exp\left\{-\beta t + \frac{\sigma^2}{2}t\right\}\right] \tag{10}$$

Similarly, we consider the sample path of fault fixing time required for OSS maintenance, e.g., the needed remaining fault fixing time from time t to the end of the project are obtained as follows:

$$\Omega_{re}(t) = \alpha\exp\{-\beta t - \sigma\omega(t)\}, \tag{11}$$

$$\Omega_{rs}(t) = \alpha(1 + \beta t)\exp\{-\beta t - \sigma\omega(t)\} \tag{12}$$

Then, the expected fault fixing time required for OSS maintenance until the end of operation time t are respectively obtained as follows:

$$\mathrm{E}[\Omega_{re}(t)] = \alpha\exp\left\{-\beta t + \frac{\sigma^2}{2}t\right\}, \tag{13}$$

$$\mathrm{E}[\Omega_{rs}(t)] = \alpha(1 + \beta t)\exp\left\{-\beta t + \frac{\sigma^2}{2}t\right\}. \tag{14}$$

Since the Wiener process $\omega(t)$ is a Gaussian process, $\log\{\alpha - \omega(t)\}$ is also a Gaussian process. The mean of $\log\{\alpha - \omega(t)\}$ are derived as follows:

$$\mathrm{E}[\log\{\alpha - \Omega_e(t)\}] = \log\alpha - \beta t, \tag{15}$$

$$\mathrm{E}[\log\{\alpha - \Omega_s(t)\}] = \log\alpha - \beta t + \log(1 + \beta t). \tag{16}$$

Also, The variance of $log\{\alpha - \omega(t)\}$ are derived as follows:

$$\mathrm{Var}[\log\{\alpha - \Omega_e(t)\}] = \sigma^2 t, \tag{17}$$

$$\mathrm{Var}[\log\{\alpha - \Omega_s(t)\}] = \sigma^2 t. \tag{18}$$

Thus, the following equation is derived:

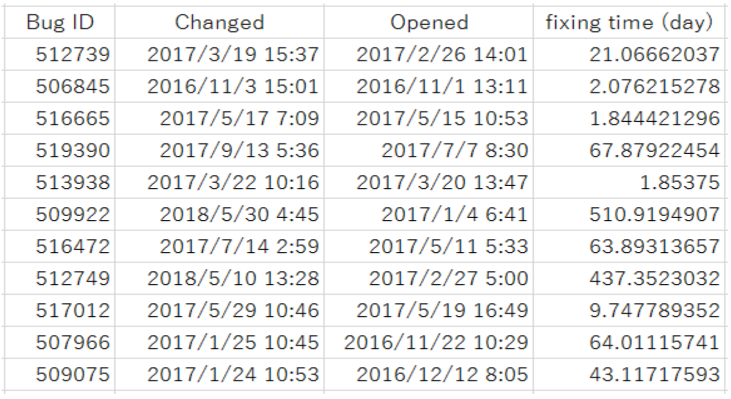
$$\Pr[\log\{\alpha - \Omega_e(t)\} \leq x] = \Phi\left(\frac{x - \log\alpha + \beta t}{\sigma\sqrt{t}}\right), \tag{19}$$

$$\Pr[\log\{\alpha - \Omega_s(t)\} \leq x] = \Phi\left(\frac{x - \log\alpha + \beta t - \log(1+\beta t)}{\sigma\sqrt{t}}\right). \tag{20}$$

where x is the cumulative fault fixing time at time t. Also, Φ means standard normal distribution and is defined as follows:

$$\Phi(x) \quad = \quad \frac{1}{\sqrt{2\sigma}}\int_{-\infty}^{x} \exp\left(-\frac{y^2}{2}\right) dy. \tag{21}$$

Considering the above points, the transition probability distributions of $\Omega_e(t)$ and $\Omega_s(t)$ are obtained as:



Bug ID	Changed	Opened	fixing time (day)
512739	2017/3/19 15:37	2017/2/26 14:01	21.06662037
506845	2016/11/3 15:01	2016/11/1 13:11	2.076215278
516665	2017/5/17 7:09	2017/5/15 10:53	1.844421296
519390	2017/9/13 5:36	2017/7/7 8:30	67.87922454
513938	2017/3/22 10:16	2017/3/20 13:47	1.85375
509922	2018/5/30 4:45	2017/1/4 6:41	510.9194907
516472	2017/7/14 2:59	2017/5/11 5:33	63.89313657
512749	2018/5/10 13:28	2017/2/27 5:00	437.3523032
517012	2017/5/29 10:46	2017/5/19 16:49	9.747789352
507966	2017/1/25 10:45	2016/11/22 10:29	64.01115741
509075	2017/1/24 10:53	2016/12/12 8:05	43.11717593

Figure 2. A part of the dataset used in this paper

$$\Pr[\Omega(t) \leq n | \Omega_e(0) = 0] = \Phi\left(\frac{\log\frac{\alpha}{\alpha - n} - \beta t}{\sigma\sqrt{t}}\right), \tag{22}$$

$$\Pr[\Omega(t) \leq n | \Omega_s(0) = 0] = \Phi\left(\frac{\log\frac{\alpha}{\alpha - n} - \beta t + \log(1+\beta t)}{\sigma\sqrt{t}}\right). \tag{23}$$

Also, the imperfect fault fixing probability is derived as follows:

$$\Pr[\Omega(t + \Delta t) \leq n | \Omega(t) = n] = \Phi\left[\left\{\log\frac{\int_0^t \beta(s)ds}{\int_0^{t+\Delta t} \beta(s)ds}\right\} / \sigma\sqrt{\Delta t}\right]. \tag{24}$$

where it is defined as the probability that n is not exceeded in a minute time interval $\Delta t (t > 0)$.

Specifically, it is possible to grasp the time when the project is converged by deriving the accumulated fault fixing time and the remaining fault fixing time. As a result, OSS user can grasp the scale of the remaining faults at a specific time. In addition, the project manager and the developer can grasp the amount of effort required in the future, and it will help to decide on future development policies. The transition probability distribution shows the probability that the cumulative fixing time will be a specific value in the operating time. In other words, it is possible to grasp the future project progress by considering an arbitrary cumulative fixing time as the transition probability distribution. It will be useful for OSS users to understand the required total cumulative fixing time α, because it is beneficial for OSS users to select the high-reliable OSS. Then, the project managers and developers can also consider the future plan of project management based on the predictions by using the proposed method.

The high probability of imperfect fault fixing means that more fault fixing time is required due to the fault fixing. In other words, it is useful for OSS users, project managers, and developers to understand the optimal time in terms of the probability of imperfect fault fixing, because the project is unstable in this case.

By using above equations, we can evaluate the stability of open source projects by deriving the accumulated fault fixing time, the remaining fault fixing time, the transition probability distribution of the proposed models, and the probability of imperfect fault fixing.

Table 1. Parameter estimation in Eclipse

		Exponential	delayed S-shaped
parameter	α	1.621×10^4	1.158×10^4
	β	9.152×10^{-3}	3.535×10^{-2}
	σ	5.248×10^{-3}	1.603×10^{-2}
AIC		1274.079	1337.218

3. Application of Proposed Method to Actual Data

We discuss the applicability as a method for evaluating the stability of a project by applying actual open source project data to the proposed model.

3.1 Used Data Set

In this paper, we used one open source project to evaluate the proposed model and stability the project. For applying the proposed model to actual project data, we use the data of Eclipse obtained from Bugzilla. Eclipse is one of the open source software for integrated development environment used in computer programming. This project uses Bugzilla as open source bug tracking system. The information about reported faults is freely available from the bug tracking system. Figure 2 shows a part of the Eclipse data used in this paper. The fault fixing time used in this paper is the difference between the time when the fault was reported (Opened) and the time when the fault information was changed (Changed). In particular, we use Eclipse version 4.7 (Oxygen) data. From the data used in this research, the Eclipse project has 1760 fault fixing data, so this project is regarded as large-scaled project. Especially, we used only for fixed faults as the number of fix faults. In this paper, we apply the remaining fault fixing time, imperfect fault fixing probability and transition probability distribution derived in Section 3 to Eclipse project data. Also, we have estimated the parameters by the method of maximum likelihood.

In this paper, the unit of fault fixing time is "day", and the fault fixing time represents the weekly average time. Table 1 shows the results of parameter estimation and AIC (Akaike's Information Criterion). In terms of AIC, the exponential model fits better than the delayed S-shaped model for model. Also, we mainly focus on the result of exponential model, because of AIC.

3.2 Remaining Fault Fixing Time

Figures 3~4 show the remaining fault fixing time of Eclipse project using exponential model and delayed S-shaped model in Eqs. (11)~(14). From these figures, the delayed S-shaped model estimates become lower than exponential model in terms of the fault fixing time. In other words, the delayed S-shaped model estimates fault fixing time pessimistically. In addition, we can judge that project is not stable because the delayed S-shaped model expresses noise largely. By using this result, it is possible to know when the project will converge. Figures 3~4 show that the remaining fault fixing time has converged to 0.

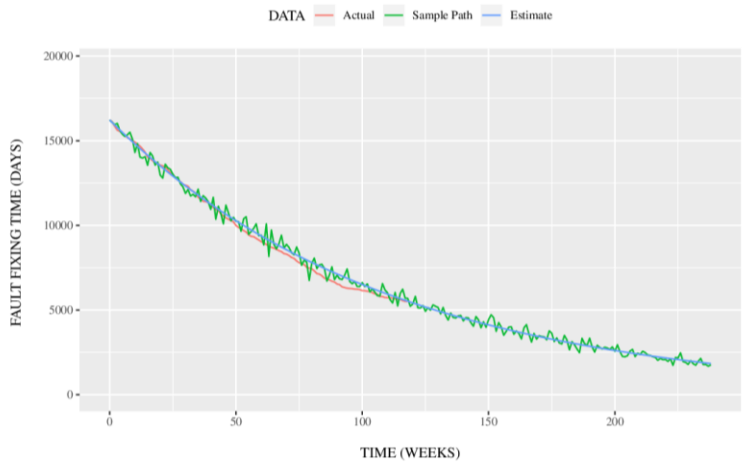


Figure 3. Remaining fault fixing time using exponential model in Eqs. (11) and (13)

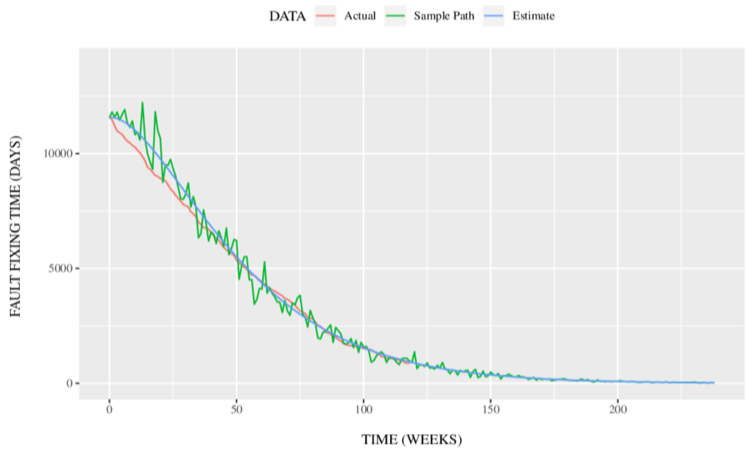


Figure 4. Remaining fault fixing time using delayed S-shaped model in Eqs. (12) and (14)

3.3 Transition Probability Distribution and Imperfect Fault Fixing Probability

Figures 5~6 show the fault fixing time transition probability distribution in Eq. (22). Figure 5 shows the result of substituting actual data for x in Eq. (22). In other words, the value of x differs every time. In conclusion, we calculate the probability of the actual value in the prediction model. Figure 6 substitutes fixed values for x in Eq. (22). We can roughly estimate the time to reach any value x.

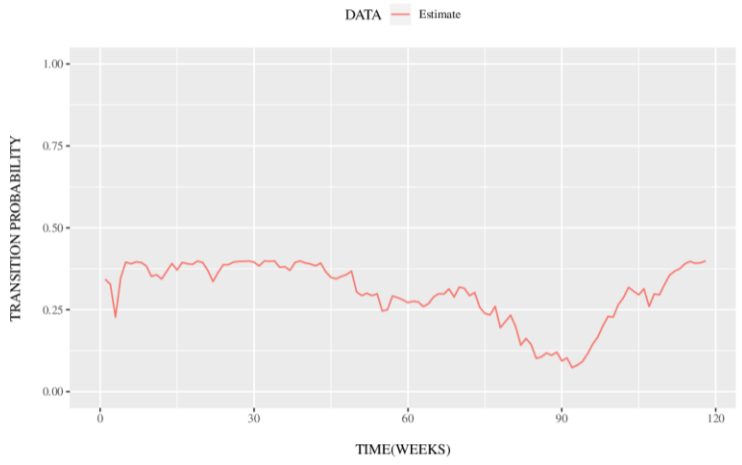


Figure 5. Fault fixing time transition probability distribution using exponential model in Eq. (22) with actual data

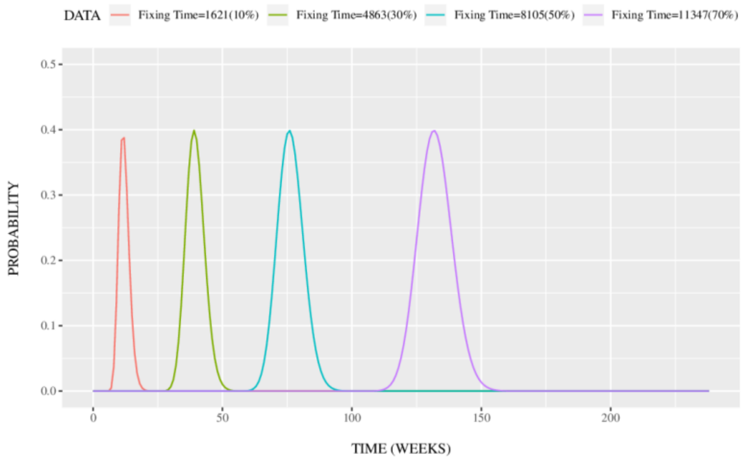


Figure 6. The estimated transition probability distribution using exponential model in Eq. (22)

Especially, Figure 5 shows relationship between accumulated fixing time $\Omega(t)$ predicted by parameter estimation and actual open source project data. By highly estimating this probability, we can judge that the cumulative fixing time $\Omega(t)$ predicted by parameter estimation is predicted lower than the actual data. Also, Figure 6 shows the probability of reaching a certain cumulative fixing time at time t. In particular, Figure 6 shows the probability distribution of the time to reach 10%, 30%, 50% and 70% of the estimated cumulative fixing time α. By using this result, it is possible to grasp the time when the project converges. Therefore, for the user who considers the risk due to the occurrence of the fault, the problem can be solved by specifying the time when no further fault fixing is required. In this case, the probability distribution of the time to reach near 100% of the estimated cumulative fixing time α should be calculated.

Finally, Figure 7 shows the results of sensitivity analysis of the probability of imperfect fault fixing using exponential model. We can judge that the smaller the value of β, the more imperfect fault fixing frequently occurs. In particular, the probability was constant using exponential model. Therefore, we can judge that the imperfect fault fixing probability always occurs with a certain probability.

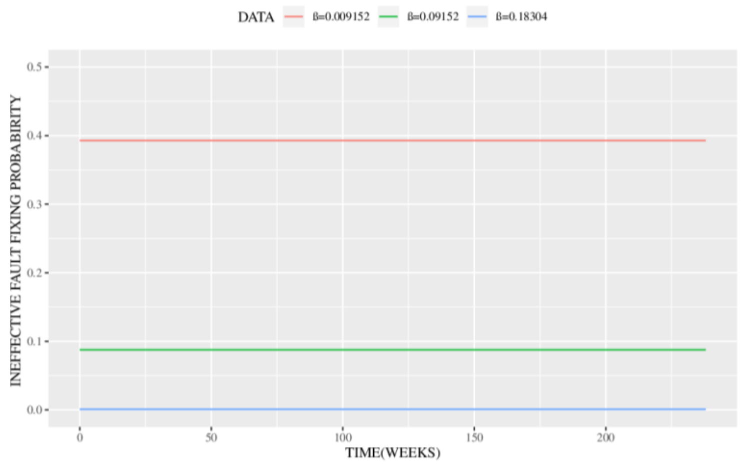


Figure 7. Imperfect fault fixing probability using exponential model in Eq. (24)

By using these indicators, open source project managers and OSS users will be able to make various decisions.

4. Conclusions

In general, the prediction of development effort and fixing time for individual faults can assess by using conventional OSS reliability evaluation methods. However, there is no researches in terms of the estimation of the fault fixing time for a long time. It is difficult to use the conventional software reliability growth model for the fault fixing time, because the conventional software reliability growth models mainly evaluate the number of faults in software development. We can easily control open source projects if we can assess the stability and reliability of future projects by using fault fixing time data. Thereby, the proposed method will lead to assess the stability of OSS systems developed under open source projects affected by various people and the environment. Also, the appropriate control of management effort for OSS will be indirectly linked to the quality, safety, reliability, and cost reduction of OSS if the manager grasps the future of the project progress.

In this paper, we discuss the stability assessment method of open source project in terms of fault fixing time based on stochastic differential equation model in open source project. In particular, considering the characteristics of changes in the fault fixing time in large-scale open source projects and the complexity in OSS development, we have derived the transition probability distribution and imperfect fault fixing probability of open source project based on Wiener process. Also, we have shown some examples using the actual open source project data for assessing the project stability. As a result, the OSS project managers and OSS users can assess the project stability in terms of fault fixing time. The proposed method helps project managers and OSS users as an evaluation method of open source project progress in the operation phase.

Conflict of Interest
The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgements
The authors would like to express their sincere thanks to the editor and anonymous reviews for their time and valuable suggestions.

References

Akbarinasaji, S., Caglayan, B., & Bener, A. (2018). Predicting bug-fixing time: a replication study using an open source software project. *Journal of Systems and Software*, *136*, 173-186.

Arnold, L. (1974). *Stochastic differential equations: theory and applications*. John Wiley & Sons, New York.

Bougie, G., Treude, C., German, M.D., & Storey, A.M. (2010). A comparative exploration of free BSD bug lifetimes. Proceedings of the *7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, Cape Town, South Africa, 2-3 May, pp.106-109.

Giger, E., Pinzger, M., & Gall, H. (2010). Predicting the fix time of bugs. Proceedings of the *2nd International Workshop on Recommendation Systems for Software Engineering*, Cape Town, South Africa, 4-4 May, pp.52-56.

Hooimeijer, P., & Weimer, W. (2007, November). Modeling bug report quality. Proceedings of the *Twenty-Second IEEE/ACM International Conference on Automated Software Engineering* (pp. 34-43). https://doi.org/10.1145/1321631.1321639.

Kapur, P.K., Pham, H., Gupta, A., & Jha, P.C. (2011). *Software reliability assessment with OR applications*. Springer-Verlag, London.

Kumar, R., Kumar, S., & Tiwari, S.K. (2019). A study of software reliability on big data open source software. *International Journal of System Assurance Engineering and Management*, *10*(2), 242-250.

Lyu, M.R. (1996). *Handbook of software reliability engineering*. IEEE Computer Society Press; Los Alamitos.; CA, U.S.A.

Musa, J.D., Iannino, A., & Okumoto, K. (1987). *Software reliability: measurement, prediction*. Application. McGraw-Hill; New York.

Raymond, S.E. (1999). *The cathedral and the bazaar: musings on linux and open source by an accidental revolutionary*. O'Reilly and Associates, Sebastopol, California.

Sun, C., Lo, D., Wang, X., Jiang, J., & Khoo, S.A. (2010). A discriminative model approach for accurate duplicate bug report retrieval. Proceedings of the *32nd ACM/IEEE International Conference on Software Engineering (ICSE'10)*, Cape Town, South Africa. 2-8 May, pp.45-54.

The Eclipse Foundation, The Eclipse Project, http://www.eclipse.org/

Wong, E., & Hajek, B. (1981). *Stochastic processes in engineering systems*. McGraw Hill; New York.

Yamada, S. (2014). *Software reliability modeling: fundamentals and applications* (Vol. 5). Tokyo: Springer.