Scientific Research Publishing

Journal of Software Engineering and Applications, 2022, 15, 183-196
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

A Study of Quantitative Progress Evaluation Models for Open Source Projects

Hironobu Sone[1], Yoshinobu Tamura[1], Shigeru Yamada[2]

[1]Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Yamaguchi, Japan
[2]Graduate School of Engineering, Tottori University, Tottori, Japan
Email: c002wcw@yamaguchi-u.ac.jp, tamuray@yamaguchi-u.ac.jp, yamada@tottori-u.ac.jp

How to cite this paper: Sone, H., Tamura, Y. and Yamada, S. (2022) A Study of Quantitative Progress Evaluation Models for Open Source Projects. *Journal of Software Engineering and Applications*, **15**, 183-196.
https://doi.org/10.4236/jsea.2022.155010

Received: April 27, 2022
Accepted: May 28, 2022
Published: May 31, 2022

Abstract

Open source software (OSS) has become an indispensable part of society, not only for personal use but also for corporate use. Projects developed and operated by OSS are called open source projects, and the number of such projects is increasing. On the other hand, because anyone can participate in an open source project, the progress of the project is uncertain due to differences in project members' skills, development environments, and time zones of activity. Therefore, many users and companies need to understand the development and operation status of open source project. Then, the developers carefully make decisions on upgrading or installing new OSS. In this paper, we focus on the maintenance effort estimation for open source projects considering uncertainty. Also, we evaluate the project quantitatively using Earned Value Management (EVM). Moreover, we examine the appropriateness of the model for predicting the maintenance effort expeditures. Furthermore, we discuss the appropriateness of this EVM method.

Keywords

Open Source Software, Stochastic Differential Equation, Earned Value Management, Software Reliability Growth Model

1. Introduction

Open source software (OSS) is code-designed to be accessible to everyone. OSS can be viewed, modified, and distributed as desired by anyone. It is often cheaper, more flexible, and more long-lasting than proprietary software, because OSS is developed by an open source community rather than a single author or company. However, because anyone can join an open source community, there is a high degree of uncertainty about project progress due to differences in project members' skills, development environments, and time frames of activity. There-

fore, it is difficult to predict the progress in open source projects, and many users and companies need to understand the development and operation status of open source projects in terms of making decisions on upgrading or installing of OSS.

Such issues have led to researches on progress forecasting in open source projects [1] [2] [3] [4]. Many researches of project progress forecasting include the estimating of the required effort in order to resolve the reported faults [1] [2] [3]. Moreover, there are several research papers in order to estimate the maintenance effort required by individual developers [4].

On the other hand, there are few researches that estimate the amount of maintenance effort for the entire project and evaluate the stability of the project. Tamura *et al.* [5] have researched time-series prediction of maintenance effort for an entire open source project, but their evaluation of the project's progress is limited because it is limited to simple effort prediction.

In this paper, we examine a method for evaluating project stability based on maintenance effort in open source projects. In particular, we use software reliability growth models [6] [7] [8] [9] to predict the number of maintenance effort for open source projects considering uncertainty. For example, there is the stochastic approach based on stochastic differential equation for the other research area [10]. Then, we evaluate the project stability and quantitatively by using earned value management (EVM) [11]. Finally, we discuss the appropriateness of the model used in predicting maintenance effort, and discuss the appropriate model for this method.

2. Evaluation Approach for Open Source Project Stability

2.1. EVM: Overview

In this paper, we use an EVM methodology for the stability evaluation for open source project. The EVM is one of the project management methodology for measuring the project performance and progress. The project progress evaluation by using EVM is used not only for software development but also for open source projects in various fields.

We can grasp the current cost and schedule condition in the project by using the EVM. The EVM basically measures the project progress and performance by using three indices: Earned Value (EV), Planned Value (PV), and Actual Cost (AC) as shown **Figure 1**. Also, we can quantitatively grasp the current status of the project by comparing three indices such as **Table 1**.

Sone *et al.* [12] have researched the applicability of EVM to open source projects was verified, However, PV could not be derived due to a difficulty with the method used to derive the effort. In this paper, we try to derive EVM indices properly.

2.2. SRGM: Overview

In the testing process of software development, the number of potential faults in

the software decreases with the progress of testing time, because a lot of resources are spent on fault detection and correction. Therefore, the probability of software fault occurrence decreases with the testing time. Then, the software reliability and the interval of software fault occurrence time increase. Such software reliability model describes software fault phenomenon. This is called software reliability growth model (SRGM).

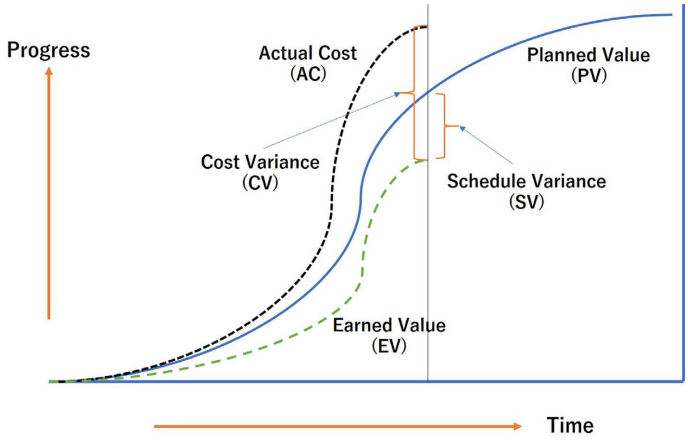

Figure 1. An example of EVM.

Table 1. Several examples of the indices used in EVM.

EVM Elements	Explanation
Planned Value (PV)	total budget estimated at the planning phase up to a certain point
Earned Value (EV)	total budgeted cost resources of processes completed at a certain point
Actual Cost (AC)	total actual cost resources invested
Budget at Completion (BAC)	total budget to completion defined at the time of planning
Cost Variance (CV)	shows whether a project is under or over budget CV = EV − AC
Cost Performance Index (CPI)	evaluates how efficiently the project is using its resources. CPI = EV/AC
Schedure Variance (SV)	determines whether a project is ahead of or behind schedule. SV = EV − PV
Schedule Performance Index (SPI)	evaluates how efficiently the project team is using its time. SPI = EV/PV
Estimate at Completion (EAC)	final cost of the project in case of continuing current performance trend. EAC = BAC/CPI
Estimate to Complete (ETC)	shows what the remaining work will cost. ETC = (BAC − EV)/CPI

In this paper, we use three models such as the exponential model, the delayed S-shaped model, and the infection S-shaped model. These are well-known models in the SRGM. We apply these models to estimate the open source projects by using EVM.

2.3. Effort Prediction Modeling for Open Source Projects

Considering the characteristic of the operation phase in open source projects, the time-dependent expenditure behavior of maintenance effort keeps an irregular state in the operation phase, because there is variability among the levels of project members' skill. Then, the time-dependent effort expenditure behavior of operation phase becomes unstable.

The operation phases of many open source projects are influenced from the external factors by triggers such as the difference of skill, and the time lag of development and maintenance activities. Considering the above points, we apply stochastic differential equation modeling for managing of the open source project. Then, let $\Omega(t)$ be the cumulative maintenance effort expeditures, such as finding software faults and improving functionality up to operational time $t(t \geq 0)$ in the open source project. Suppose that $\Omega(t)$ takes on continuous real values. $\Omega(t)$ gradually increases as the operational procedures go on. Based on SRGM approach [6] [7], the following linear differential equation in terms of the maintenance expence effort can be formulated as:

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

where $\beta(t)$ is the increase rate of maintenance effort at operational time t and a non-negative function, and α means the estimated maintenance effort expenditures required until the end of operation.

Therefore, we extend Equation (1) to the following stochastic differential equation with Brownian motion [13]:

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

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 [14], we can obtain the solution of Equation (2) under the initial condition $\Omega(0) = 0$ as follows:

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

where $\omega(t)$ is the 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 maintenance effort in case of $\beta(t)$ defined as [15]:

$$\int_0^t \beta(s)\mathrm{d}s \doteq \frac{\dfrac{\mathrm{d}F_*(t)}{\mathrm{d}t}}{\alpha - F_*(t)}. \tag{4}$$

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

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

$$F_s(t) \equiv \alpha\left\{1-(1+\beta t)\mathrm{e}^{-\beta t}\right\}, \tag{6}$$

$$F_i(t) \equiv \frac{\alpha\left\{1-\exp(-\beta t)\right\}}{1+c\cdot\exp(-\beta t)}, \tag{7}$$

where $\Omega_e(t)$ means the cumulative maintenance effort expenditures for the exponential software reliability growth model with $F_e(t)$. Similarly, $\Omega_s(t)$ is the cumulative maintenance effort expenditures for the delayed S-shaped software reliability growth model with $F_s(t)$. Also, $\Omega_i(t)$ means the cumulative maintenance effort expenditures for the inflection S-shaped software reliability growth model with $F_i(t)$, respectively.

Therefore, the cumulative maintenance effort, Ω_e, Ω_s and Ω_i up to time t are obtained as follows:

$$\Omega_e(t) = \alpha\left[1-\exp\left\{-\beta t-\sigma\omega(t)\right\}\right], \tag{8}$$

$$\Omega_s(t) = \alpha\left[1-(1+\beta t)\exp\left\{-\beta t-\sigma\omega(t)\right\}\right], \tag{9}$$

$$\Omega_i(t) = \alpha\left[1-\frac{1+c}{1+c\cdot\exp(-\beta t)}\exp\left\{-\beta t-\sigma\omega(t)\right\}\right]. \tag{10}$$

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

$$\mathrm{E}\left[\Omega_e(t)\right] = \alpha\left[1-\exp\left\{-\beta t+\frac{\sigma^2}{2}t\right\}\right], \tag{11}$$

$$\mathrm{E}\left[\Omega_s(t)\right] = \alpha\left[1-(1+\beta t)\exp\left\{-\beta t+\frac{\sigma^2}{2}t\right\}\right], \tag{12}$$

$$\mathrm{E}\left[\Omega_i(t)\right] = \alpha\left[1-\frac{1+c}{1+c\cdot\exp(-\beta t)}\exp\left\{-\beta t+\frac{\sigma^2}{2}t\right\}\right]. \tag{13}$$

2.4. Derivation of EVM for Open Source Project

In EVM for open source project, the period of data used for Planned Value (PV) and Actual Cost (AC) have the different values. Both PV and AC use the data obtained from the bug tracking system and required by the fault reporters and the fault correctors. In the open source projects, we assume that the project period is from OSS release to EOL (End of Life). Then, we can use the maintenance effort data until OSS release based on Equations (8)-(13) in order to derive PV. In particular, the parameter α in Equations (8)-(13) mean as the estimated

maintenance effort at the time *t*, when OSS is released. Therefore, the parameter α can be rephrased as Budget at Completion (BAC) in EVM. AC uses the maintenance effort data including after the OSS release. Therefore, the start time of the data used to derive PV and AC is the same.

Earned Value (EV) is the cumulative maintenance effort expeditures viewed on the same scale as the project budget (BAC). Therefore, if the OSS development effort increases but the fault is not resolved, the value of EV becomes small. Then, it is regarded as an inefficient open source project. In the derivation of EV value, the number of potential faults predicted from the fault data reported up to the time of OSS release is used. We use Equations (8)-(13) to predict the number of potential faults. We derive the fault resolving cost, *i.e.*, the value obtained by dividing the number of potential faults from the BAC, as follows:

$$\gamma = \frac{\mathrm{BAC}}{p}. \tag{14}$$

Then, γ means the fault resolving cost, and p means the potential faults at OSS release. We can derive the EV in cases of $F_e(t)$, $F_s(t)$, and $F_i(t)$ by using the fault resolving cost γ and the cumulative number of resolved faults up to the operating time *t*.

$$EV_e(t) = \gamma\left[\alpha_f\left[1-\exp\left\{-\beta_f t-\sigma_f\omega(t)\right\}\right]\right], \tag{15}$$

$$EV_s(t) = \gamma\left[\alpha_f\left[1-\left(1+\beta_f t\right)\exp\left\{-\beta_f t-\sigma_f\omega(t)\right\}\right]\right], \tag{16}$$

$$EV_i(t) = \gamma\cdot\alpha_f\left[1-\frac{1+c_f}{1+c_f\cdot\exp\left(-\beta_f t\right)}\exp\left\{-\beta_f t-\sigma_f\omega(t)\right\}\right]. \tag{17}$$

Then, α_f, β_f, c_f, and σ_f are parameters used to predict the cumulative number of resolved faults at time *t*. Therefore, the expected EV required for OSS maintenance until the end of operation time *t* are respectively obtained as follows:

$$\mathrm{E}\left[EV_e(t)\right] = \gamma\cdot\alpha_f\left[1-\exp\left\{-\beta_f t+\frac{\sigma_f^2}{2}t\right\}\right], \tag{18}$$

$$\mathrm{E}\left[EV_s(t)\right] = \gamma\cdot\alpha_f\left[1-\left(1+\beta_f t\right)\exp\left\{-\beta_f t+\frac{\sigma_f^2}{2}t\right\}\right], \tag{19}$$

$$\mathrm{E}\left[EV_i(t)\right] = \gamma\cdot\alpha_f\left[1-\frac{1+c_f}{1+c_f\cdot\exp\left(-\beta_f t\right)}\exp\left\{-\beta_f t+\frac{\sigma_f^2}{2}t\right\}\right]. \tag{20}$$

Then, the resolved cumulative number of faults is counted when the fault status is Closed in the bug tracking system.

In this paper, EVM uses the dataset obtained from bug tracking system to derive PV, AC, and EV. We assume the following terms in the **Table 2** as the EVM in the open source project considering the derivation of these EVM indices.

Table 2. Explanation for EVM used in this research.

EVM elements	Explanatory
Planned Value (PV)	Cumulative maintenance effort as planned value up to operational time t
Earned Value (EV)	Cumulative maintenance effort up to operational time t viewed on the same scale as BAC
Actual Cost (AC)	Cumulative maintenance effort up to operational time t in actual
Budget at Completion (BAC)	Total budget at the end of open source project
Cost Variance (CV)	$\mathrm{E}\left[CV_*(t)\right]$ obtained from EV-AC
Cost Performance Index (CPI)	$\mathrm{E}\left[CPI_*(t)\right]$ obtained from EV/AC
Schedure Variance (SV)	SV obtained from EV-PV
Schedule Performance Index (SPI)	SPI obtained from EV/PV
Estimate at Completion (EAC)	$\mathrm{E}\left[EAC_*(t)\right]$ obtained from BAC/CPI
Estimate to Complete (ETC)	$\mathrm{E}\left[ETC_*(t)\right]$ obtained from (BAC − EV)/CPI

3. Numerical Examples

3.1. Data Set

In this paper, we use the data set of open source project for deriving EVM indices. For applying the proposed model to actual project data set, we use the data of LibreOffice [16] obtained from Bugzilla. LibreOffice is an office suite OSS provided by The Document Foundation. In particular, the effort and fault data have been obtained from Bugzilla are version 7.2 for estimating PV and AC. In this paper, the cumulative number of reported faults are 298 and 878, respectively. In particular, we use the project data for about 39 weeks, before LibreOffice was released for estimating PV. For estimating AC, we also use project data for about 112 weeks. Also, each unit data is weekly.

3.2. Estimation of EVM Indices

In this section, we estimate the model parameters of the three SRGM models for estimating the maintenance effort and the number of faults in case of LibreOffice version 7.2 project. Also, we compare the appropriateness of our model with appropriate models.

Table 3 shows the results of parameter estimation of maintenance effort, and AIC (Akaike's Information Criterion) for comparison of model equations. Also, the parameter α in the PV data can be rephrased as BAC. In terms of AIC, the delayed S-shaped model is the best one for PV estimation. Figure 2 shows the results of applying the delayed S-shaped model to the open source project data.

Next, Table 4 shows the results of parameter estimation of AC, and AIC. Also, the parameter α can be rephrased as the project's estimated AC. In terms of AIC, the delayed S-shaped model is the best one for AC estimation. Figure 3 shows the results of applying the delayed S-shaped model to the open source project data.

Table 3. Parameter estimation of maintenance effort in terms of PV.

		Planned Value		
		exponential model	delayed S-shaped model	infection S-shaped model
parameter	α	2.867×10^{5}	9.283×10^{5}	1.859×10^{6}
	β	3.327×10^{-2}	2.293×10^{-2}	3.352×10^{-2}
	c	-	-	1.984×10^{1}
	σ	2.361×10^{-2}	7.504×10^{-4}	7.577×10^{-4}
AIC		798.191	**647.075**	709.147

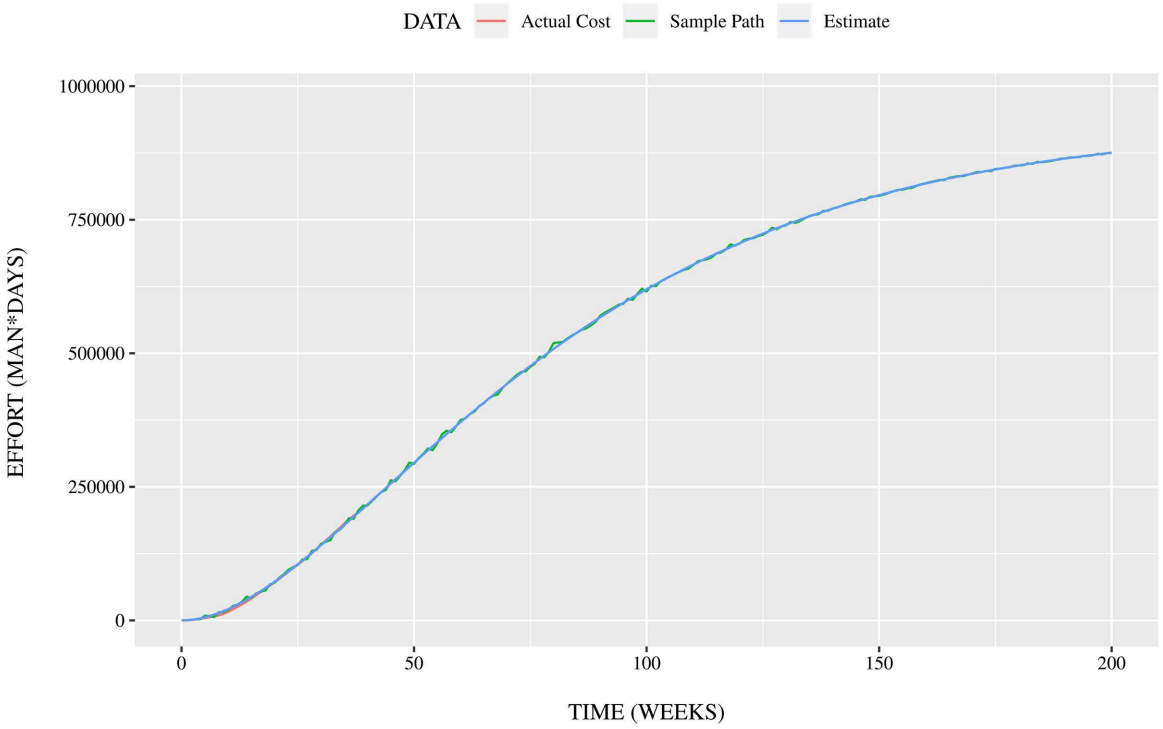

Figure 2. The cumulative maintenance effort expeditures as PV in LibreOffice Ver. 7.2 project by using Equations (9) and (12).

Table 4. Parameter estimation of maintenance effort in terms of AC.

		Actual Cost		
		exponential model	delayed S-shaped model	infection S-shaped model
parameter	α	1.325×10^{6}	1.317×10^{6}	2.728×10^{6}
	β	9.286×10^{-3}	2.387×10^{-2}	1.253×10^{-2}
	c	-	-	3.283×10^{1}
	σ	4.840×10^{-3}	2.762×10^{-3}	1.011×10^{-3}
AIC		1229.007	**1163.817**	1205.231

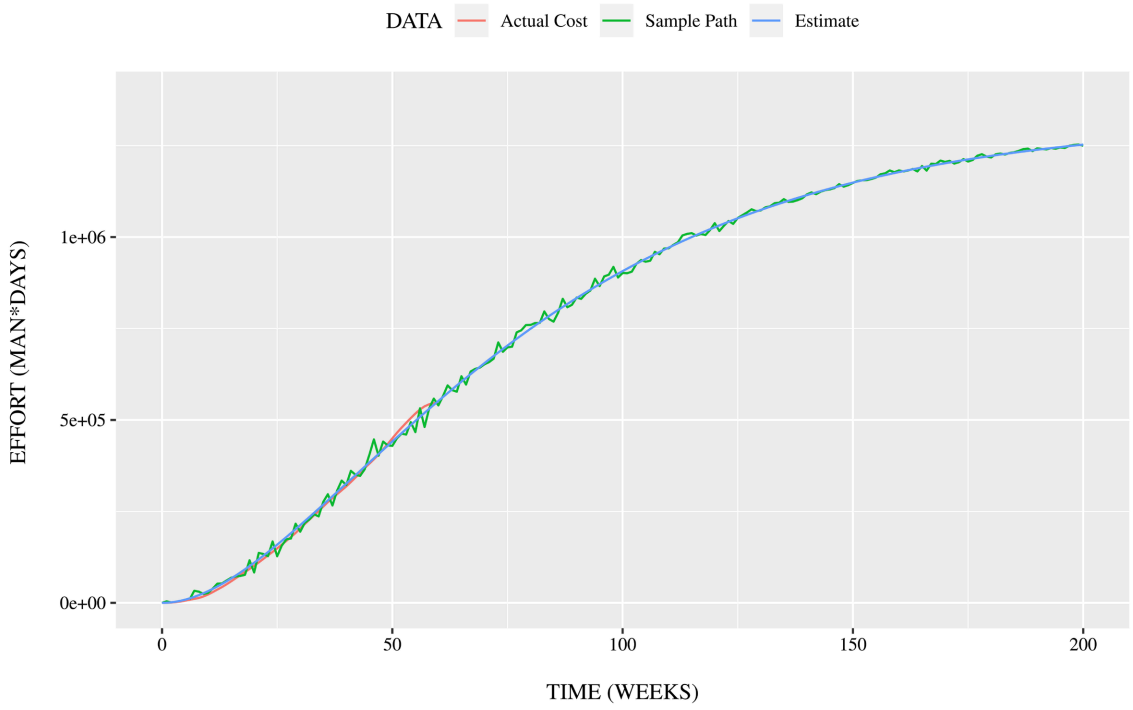

Figure 3. The cumulative maintenance effort expeditures as AC in LibreOffice Ver. 7.2 project by using Equations (9) and (12).

Also, Table 5 shows the results of parameter estimation of the estimated number of potential faults at OSS release, and AIC. We use the parameter α for deriving fault resolving cost. There is no significant difference in AIC values among all the model equations used in this research. Therefore, it is difficult for us to identify a suitable model for the data used in this research. For convenience, we assume that the exponential model with the smallest AIC is the appropriate model. Figure 4 shows the results of applying the exponential model to the open source project data.

Finally, Table 6 shows the results of parameter estimation of the estimated number of resolved faults at present, and AIC. In terms of AIC, the infection S-shaped model is the best method for the number of resolved faults estimation. Figure 5 shows the results of applying the delayed S-shaped model to the open

source project data.

A comparison of the AIC values during parameter estimation in the three model equations showed that the delayed S-shaped model and the infection S-shaped model are appropriate. In the LibreOffice version 7.2 project data, the increase rate of maintenance man-hours and number of faults at the start of the maintenance phase is small. We find that the delayed S-shaped model and infection S-shaped model are appropriate for such project data.

In open source projects, the number of fault reports increases as the number of OSS users increases after the release of a particular version. As a result, the effort required for fault maintenance increases. Therefore, the appropriate model equation for many open source project data would be the same as in this research.

Table 5. Parameter estimation of number of potential faults in case of LibreOffice.

		Estimated number of potential faults at OSS release		
		exponential model	delayed S-shaped model	infection S-shaped model
parameter	α	1.401×10^{3}	5.291×10^{2}	4.792×10^{2}
	β	6.274×10^{-3}	4.818×10^{-2}	4.718×10^{-2}
	c	-	-	2.083×10^{1}
	σ	3.165×10^{-3}	1.113×10^{-2}	1.535×10^{-2}
AIC		**238.479**	**238.613**	**238.961**

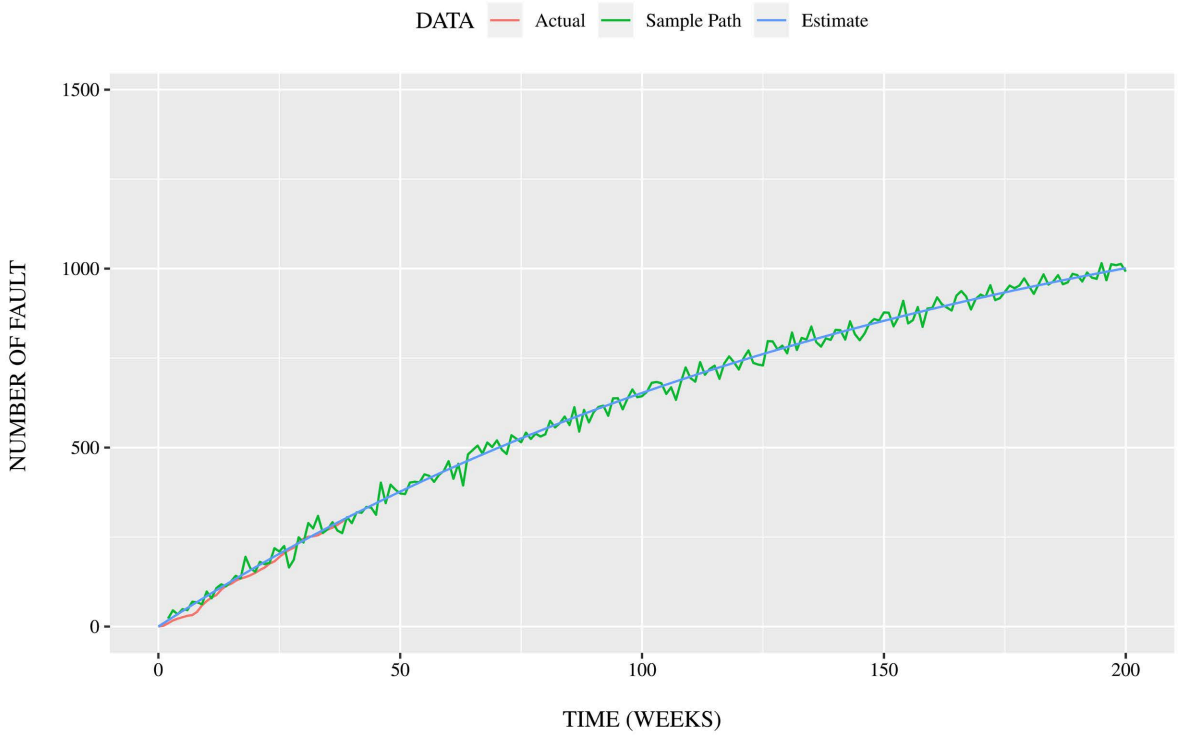

Figure 4. The cumulative estimated number of potential faults by using Equations (8) and (11).

Table 6. Parameter estimation of number of resolved faults in case of LibreOffice.

		Estimated number of resolved faults at present		
		exponential model	delayed S-shaped model	infection S-shaped model
parameter	α	1.019×10^{4}	7.887×10^{4}	6.054×10^{3}
	β	1.501×10^{-3}	2.767×10^{-3}	7.143×10^{-2}
	c	-	-	4.350×10^{2}
	σ	1.697×10^{-3}	1.395×10^{-4}	1.595×10^{-3}
AIC		534.055	488.803	**460.932**

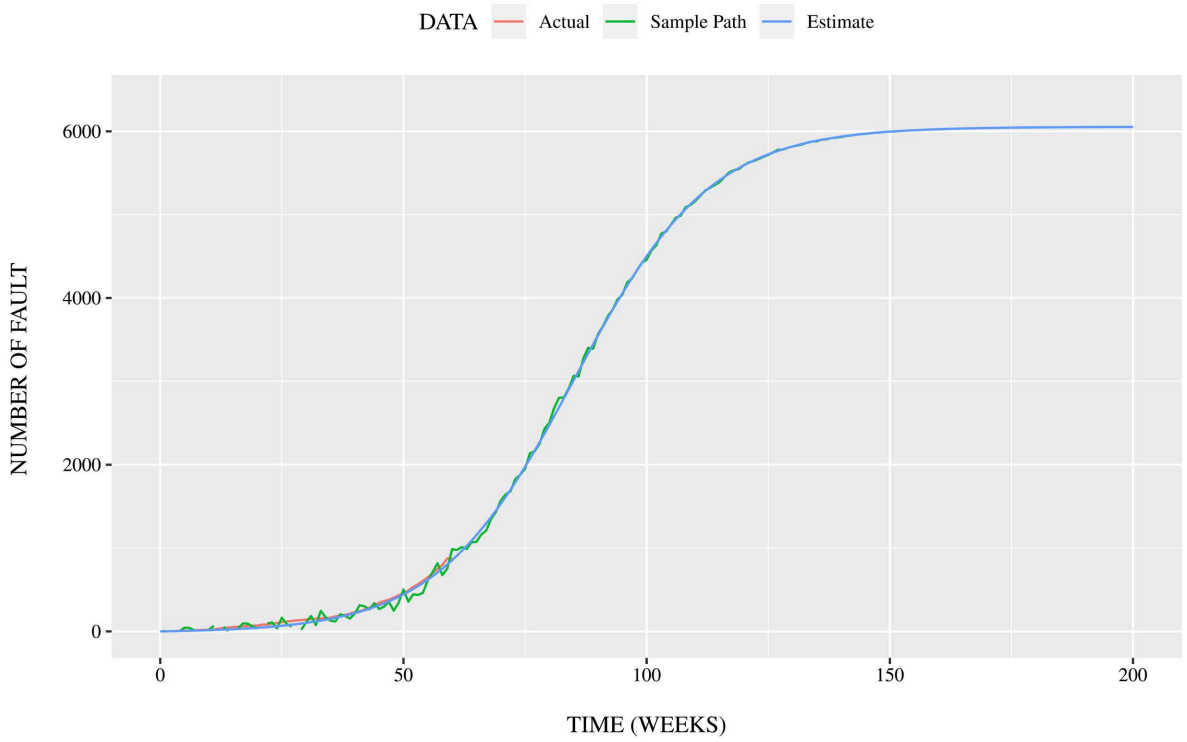

Figure 5. The cumulative estimated number of resolved faults by using Equations (10) and (13).

In this research, we derive EVM indices by using the best-fit model equation for each data set. The fault resolving cost $\gamma = 662.419(\text{man} \cdot \text{days})$, one of the EVM indices, is necessary for the derivation of EV. **Figure 6** shows the results of EV, AC, and PV estimations.

Figure 6 shows that both EV and AC are larger than PV. In particular, the EV value is very large. This is because the number of resolved faults is estimated to be higher than the number of potential faults. On the other hand, the EV value is lower than the EV and AC values around 50 weeks, the time of the version 7.2 release, showing the project is in a delayed state. In other words, after version 7.2 was released, we find that the project became more active as the number of users of that version increased.

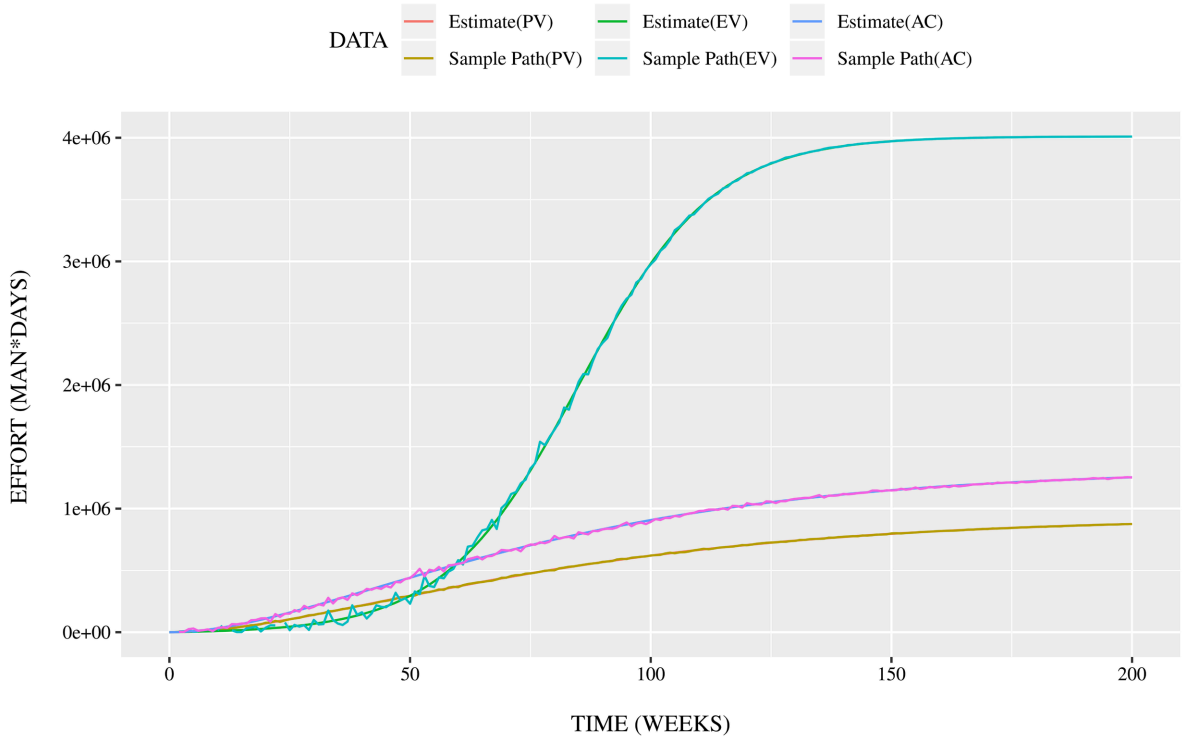

Figure 6. EVM estimation results in LibreOffice project.

4. Conclusions

In this paper, we have examined a method for evaluating project stability based on SRGM in open source projects. In terms of AIC, we have identified the appropriateness models in the open source project. Then, we have found that the delayed S-shaped model and the infection S-shaped model are the best models. We have concluded that the results are the same as other open source project, because of the characteristic of the number of OSS users' transitions. Also, we have derived EVM by using the appropriate SRGM models. As a result, we have found that the trigger for activating open source projects is after the release of a particular version.

Researches on stability evaluation methods for open source projects have often focused on the resolving of individual faults. Therefore, the practical application of EVM for evaluating the stability of open source projects as a whole will contribute to the future development of OSS. On the other hand, since the proposed method evaluates stability based on the cost of the entire open source project, it is difficult to evaluate the causes of project stability in fault units. Therefore, we consider that using not only the proposed method but also individual fault-based project evaluation methods will provide a better project stability evaluation tool.

As only one open source project data set has been used in this paper, it is necessary to verify the characteristics of the trends in maintenance effort and number of faults by using multiple project data sets in the future.

Acknowledgements

This work was supported in part by the JSPS KAKENHI Grant No. 20K11799 in Japan.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Hooimeijer, P. and Weimer, W. (2010) Modeling Bug Report Quality. *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering* (*ASE*'07), Georgia, 34-43.

[2] Giger, E., Pinzger, M. and Gall, H. (2010) Predicting the Fix Time of Bugs. *Proceedings of the* 2*nd International Workshop on Recommendation Systems for Software Engineering*, Cape Town, May 2010, 52-56. https://doi.org/10.1145/1808920.1808933

[3] Marks, L., Zou, Y. and Hassan, E.A. (2011) Studying the Fix-Time for Bugs in Large Open Source Projects. *Proceedings of the* 7*th International Conference on Predictive Models in Software Engineering* (*Promise*'11), Banff Alberta, September 2011, Article No. 11. https://doi.org/10.1145/2020390.2020401

[4] Mishra, R. and Sureka, A. (2014) Mining Peer Code Review System for Computing Effort and Contribution Metrics for Patch Reviewers. *Proceedings of the* 2014 *IEEE* 4*th Workshop on Mining Unstructured Data*, Victoria, 30 September 2014, 11-15. https://doi.org/10.1109/MUD.2014.11

[5] Tamura, Y. and Yamada, S. (2017) Open Source Software Cost Analysis with Fault Severity Levels Based on Stochastic Differential Equation Models. *Journal of Life Cycle Reliability and Safety Engineering*, **6**, 31-35. https://doi.org/10.1007/s41872-017-0009-5

[6] Yamada, S. (2014) Software Reliability Modeling: Fundamentals and Applications, Springer-Verlag, Tokyo/Heidelberg.

[7] Lyu, M.R. (1996) Handbook of Software Reliability Engineering. IEEE Computer Society Press, Los Alamitos.

[8] Musa, J.D., Iannino, A. and Okumoto K. (1987) Software Reliability: Measurement, Prediction Mechanics, Application. McGraw-Hill, New York.

[9] Kapur, P.K., Pham, H., Gupta, A. and Jha, P.C. (2011) Software Reliability Assessment with OR Applications. Springer-Verlag, London. https://doi.org/10.1007/978-0-85729-204-9

[10] Trung, N. (2018) Modeling Election Problem by a Stochastic Differential Equation. *American Journal of Operations Research*, **8**, 441-447. https://doi.org/10.4236/ajor.2018.86024

[11] Fleming, Q.E. and Koppelman, J.M. (2010) Earned Value Project Management. 4th Edition, PMI, Newton Square.

[12] Sone, H., Tamura, Y. and Yamada, S. (2019) Statistical Maintenance Time Estimation Based on Stochastic Differential Equation Models in OSS Development Project. *Computer Reviews Journal*, **5**, 126-140.

[13] Wong, E. (1971) Stochastic Processes in Information and Systems. McGraw-Hill, New York.

[14] Arnold, L. (1971) Stochastic Differential Equations-Theory and Applications. John Wiley & Sons, New York.

[15] Yamada, S., Kimura, M., Tanaka, H. and Osaki, S. (1994) Axisymmetric Vortex Solution of Navier-Stokes Equation. *IEICE Transactions on Fundamentals*, **E77-A**, 109-116.

[16] LibreOffice. https://ja.libreoffice.org/