Predicting Release Time for Open Source Software based on the Generalized Software Reliability Model

Hironori Washizaki*, Kiyoshi Honda* and Yoshiaki Fukazawa*

* Global Software Engineering Laboratory, Waseda University
3-4-1 Ohkubo, Shijuku-ku, 169-8555 Tokyo, JAPAN
Email: washizaki@waseda.jp, khonda@ruri.waseda.jp, fukazawa@waseda.jp

***Abstract*—There is a significant challenge that how to predict the possible release date of the target software having enough reliability in agile development where incremental development and small software releases are key characteristics. Existing approaches targeting agile development usually use release backlogs for predicting and setting delivery windows; however these do not consider the reliability of software for release date prediction so that there is a possibility that software at the predicted release date have poor reliability. Previously we proposed a generalized software reliability model (GSRM) based on a stochastic process and compared it with other models to evaluate recent software developments. However, we, did not directly evaluate the accuracy of the predicted release time by model. In this paper, towards prediction of release dates in agile development, we focus on the release dates of open source software (OSS) developments and the number of detected issues (faults) since OSS developments comply well with the definition of the agile development in terms of incremental process and frequent releases We define the accuracy of the predicted release time using the given development terms and the number of issues. Additionally, we propose a method to evaluate the accuracy of the predicted release time. In the best case, GSRM shows only 0.572% Error Rate, which corresponds to a predicted release date of two days prior to the actual release date. We believe that our method should be applicable to agile developments too.**

I. Introduction

Agile development is aimed at minimizing overall risk and encouraging rapid and flexible response to specification changes by using iterative and incremental process model[1]. Incremental development and small software releases are key characteristics of agile software development processes. In such processes, there is a significant challenge that how to predict the possible release date of the target software having enough reliability. Existing approaches such as an approach based on Cumulative Flow Diagrams targeting agile development[2] usually use release backlogs for predicting and setting delivery windows; however these do not consider the reliability of software for release date prediction so that there is a possibility that software at the predicted release date have poor reliability.

Software reliability is a critical component of computer system availability. Software reliability growth models can be used to indicate whether enough faults have been removed to release the software. Although the logistic curve and Gompertz curve[3] are well-known software reliability growth curves, they cannot account for the dynamics of software development. Development is affected by various elements of the development environment, including the skills of the development team and changes in requirements. Especially in agile developments, there could be various dynamic situations such as changing and prioritizing requirements frequently.

Examples of software reliability models include the ”Times Between Failures Models” and ”Failure Count Models”[7]; among them we used the ”Failure Count Model,” which is based on counting failures (issues) and probability methods. The Goel-Okumoto NHPP Model and the Musa Execution Time Model are examples of this type of model [7]. Some of recent studies by Tamura[8], Yamada[9], Cai[10], Kamei[11], Dohi[7], Schneidewind[12], Nguyen[13], and Okamura[14] have attempted to describe the dynamics of developments using stochastic differential equations. Although many models have been proposed, surveyed, and compared[15], [16], [23], most failure count models cannot account for the dynamics of development, such as drastic changes in the development team composition or significant reductions in the development time. And these conventional models cannot precisely predict when developments will end.

To predict the time range that a development will end, here we propose a method employing our model, a generalized software reliability model (hereafter GSRM[5], [6]), which can describe several development situations involving random factors, such as the skills of teams and the development environment. Previous studies[17] have employed only linear stochastic differential equations, but our research indicates that non-linear stochastic differential equations lead to elaborate equations that can model situations more realistically[6].

Moreover, towards prediction of release dates in agile development, we applied our method to several versions in

[0]In our previous papers[4], [5], [6], we proposed GSRM and applied it to several versions of OSS; however we have not evaluated the results in detail. In this paper we suggest a method to predict the release time using data for six months after the first issue is detected by GSRM and another model in detail.

a certain open source software (OSS) development to reveal the development time frame of typical OSSs. As a result, we successfully predicted the release dates of versions of OSS more accurately than a representative conventional approach.

Agile developments and OSS developments share many principles and values[18]. We believe that our method should be applicable to agile developments since OSS developments comply well with the definition of the agile development in terms of incremental process and frequent releases[19], and it is indicated that all the agile methods are in essence applicable to open source software development because of their iterative and incremental character[20]. Although in some agile development methods release dates are fixed to some extent due to fixed length of iterations, our method should be still beneficial for various purposes such as predicting which releases will have enough reliability.

This paper aims to answer the following research questions.

RQ1: Is GSRM better than other models (e.g., NHPP) from the viewpoint of prediction of number of issues?

RQ2: Is GSRM better than other models (e.g., NHPP) from the viewpoint of prediction of release dates?

Our contributions are as follows.

- A two-step method to predict the release time of OSS: separation of development time periods into different versions, and, application of GSRM for prediction.
- A method to evaluate the prediction accuracy in terms of release date by defining a Error Rate as a relative amount of prediction error of the release date.
- An evaluation result targeting a OSS front-end framework ”foundation”[21] showing that our prediction method works better than NHPP.

The remainder of the paper is organized as follows. Section II describes our method, while section III evaluates our method. Section IV discusses related works. Finally, section V provides a conclusion and future direction.

II. Proposed method

In this section, we propose a two-step method to predict the release time and a method to evaluate the prediction accuracy in terms of the release time.

A. Prediction of release time

To determine when OSS can be released with respect to the number of detected faults, we propose the two-step method described below: *1)* and, *2)* Using GSRM to predict the number of faults and the release date.

1) Separation into time periods: The upper graph in Fig. 1 indicates the number of detected faults for the ”foundation,”[21] which is a OSS front-end framework, divided by each version. The curve shape is sharper when a newer version is released. Therefore, the versions are separated based on the changing points before applying our model (GSRM) because such separation allows the model to more precisely approximate the data.

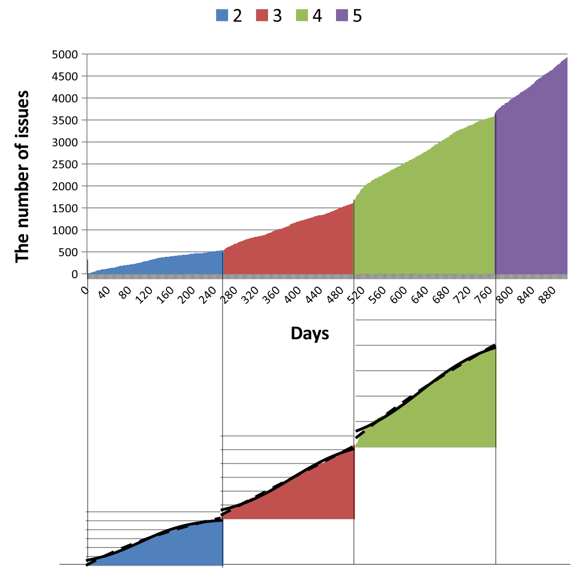

Fig. 1. Number of detected faults and development days for the ”foundation.”

2) GSRM: For our software reliability model, we extend a nonlinear differential equation that describes the fault content as a logistic curve to an Ito-type stochastic differential equation. We start with the logistic differential equation, which is expressed as

$$dN(t)/dt = N(t)(a + bN(t)) \tag{1}$$

$N(t)$ is the number of detected faults at time t, a defines the growth rate, and b is the carrying capacity. If $b = 0$, then the solutions of this equation are exponential functions. We extend equation (1) into a stochastic differential equation because actual developments do not correctly obey equation (1) due to numerous uncertainties and dynamic changes.

We consider such dynamic elements to be time-dependent and to contain uncertainty, which are expressed using a. The time-dependence of a can be used to describe situations such as the improved skills of development members and increased growth rate. The uncertainty of a can describe parameters such as the variability of development members and environment. We analyze the growth of software with an emphasis on the test phase by simulating the number of detected faults. We assume that the software development has the following properties.

- The total number of faults is constant.
- The number of faults that can be found depends on time.
- The number of faults that can be found contains uncertainty that can be simulated with Gaussian white noise.

Considering these properties, we extend equation (1) to an Ito-type stochastic differential equation with $a(t) = \alpha(t) + \sigma dw(t)$ as shown below.

$$dN(t) = (\alpha(t) + \beta N(t))N(t)dt + N(t)\sigma dw(t) \tag{2}$$

TABLE I
COMBINATIONS OF $\alpha(t)$ AND $\gamma(t)$.

	$\gamma_1(t) = N(t)\sigma dw(t)$	$\gamma_2(t) = \sigma dw(t)$	$\gamma_3(t) = 1/N(t)\sigma dw(t)$
$\alpha_1(t) = a_1$(const.)	The number of issues per unit time is constant but the uncertainty increases near the end. This model is similar to a logistic curve. (**Model 1-1**)	The number of issues per unit time is constant and the uncertainty is constant at any given time. (**Model 1-2**)	The number of issues per unit time is constant but the uncertainty is greater at the start of the project than at the end (e.g., the team matures over time).(**Model 1-3**)
$\alpha_2(t) = a_2(t < t_1)$ $\alpha_2(t) = a_3(t \geq t_1)$	The number of issues per unit time changes at t_1, and the uncertainty increases near the end (e.g., new members join the project at time t_1).(**Model 2-1**)	The number of issues per unit time changes at t_1 but the uncertainty is constant at any given time.(**Model 2-2**)	The number of issues per unit time changes at t_1 but the uncertainty is greater at the start of the project than at the end.(**Model 2-3**)
$\alpha_3(t) \propto t$	Both the number of detected faults per unit time and the uncertainty increase near the end (e.g., increasing manpower with time).(**Model 3-1**)	The number of detected faults per unit time increases but the uncertainty is constant at any given time.(**Model 3-2**)	The number of detected faults per unit time increases but the uncertainty is greater at the start of project than at the end.(**Model 3-3**)

$\alpha(t) + \sigma dw(t)$ is the differential of the number of detected faults per unit time, $\gamma(t) = N(t)\sigma dw(t)$ is the uncertainty term, σ is the dispersion, and β is the nonlinear carrying capacity term. This equation has two significant terms: α and dw. α affects the end point of development, while dw affects the growth curve through uncertainties. In particular, the stochastic term depends on $N(t)$, which means that uncertainties depend on the number of detected faults. We compare three different types of dependencies of $\gamma(t)$ on $N(t)$.

(a) $\gamma_1(t) = N(t)\sigma dw(t)$.
(b) $\gamma_2(t) = \sigma dw(t)$($\gamma(t)$ does not depend on $N(t)$).
(c) $\gamma_3(t) = 1/N(t)\sigma dw(t)$($\gamma(t)$ depends on the inverse of$N(t)$).

Table I summarizes the types of $\alpha(t)$, the coefficient of $dw(t)$, and the corresponding situations.

The table indicates that the reliability growth models can be applied to nine types of development situations. Existing models can describe only one of these situations with additional limitations. In contrast, GSRM can describe several of these situations. This is primarily because existing models cannot handle time-dependent growth rates without limitations, whereas GSRM can handle the time-dependence growth rates.

B. Evaluation of accuracy

We use GSRM to predict the release time. For comparison, we also predict the release date using the Non-Homogeneous Poisson Process (NHPP) model[22], [23] since the NHPP model is the most popular one[6]. Herein we assume that the release time is defined as the time where 95% of the maximum number of predicted issues are detected. It should be noted that this definition depends on the development team's policy.

Fig. 2 shows that the predicted release times driven by GSRM and NHPP model. Herein the predicted release time is defined as the intersection between the model line and the point where 95% of the maximum number of predicted issues is detected. Additionally, we propose a method to evaluate the accuracy of the predicted date using limited datasets of issues and dates. For example, here we define the terms as the data from the date when the first issue was detected to six months later (180 days).

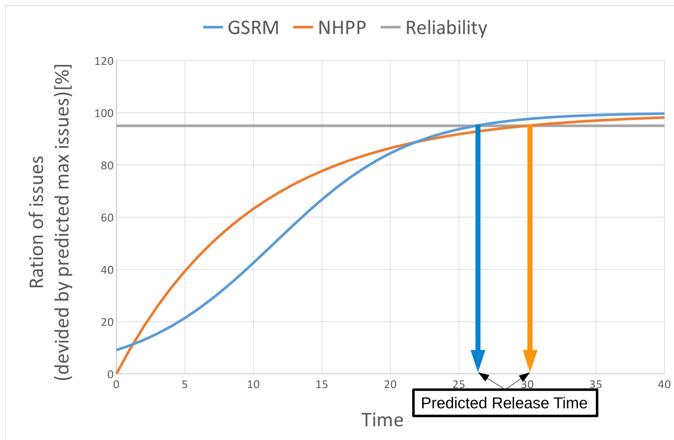

Fig. 2. Predicted release dates by model using the threshold that 95% of the maximum number of faults are detected.

Figure 3 graphically depicts this method where the data in the blue dotted line is used to predict the release time in the red box.

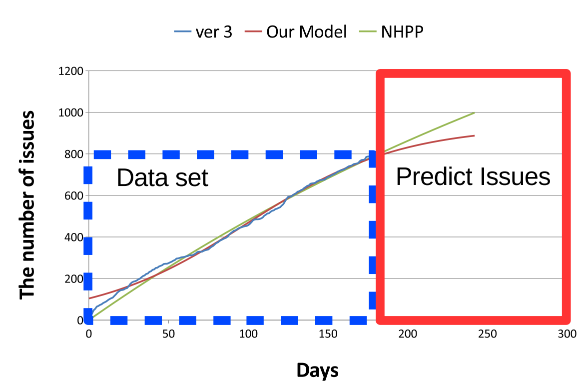

Fig. 3. Number of days until release (red line) is predicted using the data in the blue dotsdotted line, we predicted for the number of issues in the red line about "foundation."

We evaluated the prediction accuracy using the Error Rate, which is defined as

$$\mathrm{Error\,Rate} = \frac{\mathrm{Predicted\,Release\,Day} - \mathrm{Release\,Day}}{\mathrm{Release\,Day}} \quad (3)$$

In equation (3), the Error Rate means the relative amount of prediction error of the release date. The Predicted Release Day is defined as the value where the model detects 95% of the maximum number of predicted faults. The dataset in the model includes data from the date when the first issue was detected to six months later. If the Error Rate is less (greater) than 0, the model predicts a release day earlier (later) than the actual release day.

If one model precisely approximates the release date for one OSS, it may not be accurate for future values. Thus, estimating the accuracy of model predictions is important, which is why we evaluate the prediction accuracy of a model.

III. Evaluation

To answer the following research questions, we conducted a case study targeting actual development data of OSS ("foundation") in a given time-independent situation obtained from Github site[24], and evaluated our method employing GSRM and general NHPP models.

RQ1: Is GSRM better than other models (e.g., NHPP) from the viewpoint of prediction of number of issues?

RQ2: Is GSRM better than other models (e.g., NHPP) from the viewpoint of prediction of release dates?

The time limitation is necessary because the NHPP model cannot be applied to time-dependent situations. We also compare the predicted numbers of issues driven by NHPP and GSRM at the end of development using six months (180 days) of data.

As GSRM, we chose **Model 1-2** (i.e. both of the number of issues per unit time and the uncertainty are constant) among all of nine model types in Table I because of two reasons. Firstly it was hard to identify specific uncertainty type and specific type of dependencies of $\gamma(t)$ on $N(t)$. Secondly we wanted to make the comparison and discussion simple. Comparison with other model types (such as Model 1-3) is one of our future works.

A. Prediction of Number of Issues (RQ1)

Table II[5] shows the number of predicted issues, days of development for each version, the residual sum of squares (RSS), and the Akaike's Information Criterion (AIC) for each model. The results show GSRM is better than NHPP from the viewpoint of prediction as GSRM more precisely predicts the number of detected issues than NHPP although RSS and AIC of GSRM are slightly larger than those of NHPP. However, Table II cannot describe the accuracy of predictions because it shows the qualities of the models and not the release day. In the next section, we evaluate the prediction accuracy of the models.

TABLE II
Quality of GSRM and NHPP models.

		Actual Data	GSRM	NHPP
Version 2	Issue	536	526	899
	Days	258	245	854
	RSS	-	50388	25929
	AIC	-	2108	1936
Version 3	Issue	1066	1170	32555
	Days	242	306	23102
	RSS	-	182119	44708
	AIC	-	2306	1965
Version 4	Issue	1974	2203	5897
	Days	265	323	2017
	RSS	-	720089	302405
	AIC	-	2865	2634

B. Prediction of Release Date (RQ2)

Table III shows the actual release date and predicted release date by model. The results show that GSRM is superior to NHPP from the viewpoint of accuracy. Hence, error rates and predicted release of GSRM are better than those of NHPP.

TABLE III
Accuracy of predictions for GSRM and NHPP models.

Version	Release day	Model	Predicted release day	Error Rate
2	258	GSRM	212	-0.178
		NHPP	1382	4.36
3	242	GSRM	240	-0.00572
		NHPP	1294	4.35
4	265	GSRM	244	-0.0778
		NHPP	445	0.679

C. Threats to Validity

For the case study, we used the actual development data of OSS as it is so that the data could contain inappropriate issue reports or some other false elements such as duplicate reports[25] and a single issue report actually containing multiple different faults. That might affect the internal validity. As our future work, we intend to confirm the validity of the data in detail.

As a threat to external validity, we only tested our method employing GSRM with single OSS, which is insufficient to make generalizations about our method. As our future work, we intend to apply our method to other OSSs. Moreover, we only compared our method with the NHPP model; although other conventional models are similar to the NHPP model, our method should also be compared to them.

IV. Related work

Power proposed an approach for predicting and setting delivery windows based on release backlogs targeting agile development[2]; the approach does not consider the reliability of software for release date prediction so that there is a possibility that software at the predicted release date have poor reliability.

Many different types of software reliability growth models exist. Yamada et al. proposed an extended NHPP model, which is related to testing-domain[26]. The test-domain dependent model includes the notion that the tester's skills should improve by degrees; thus, skills grow over time. The test-domain dependent model adds additional assumptions to the NHPP model. However they did not confirm the approach is useful for OSS developments and/or agile developments.

Typical software reliability models use waterfall development, but Fujii et al. developed a quantitative software reliability assessment method based on the familiar non-homogeneous Poisson processes for incremental development processes[27]. Fujii et al. employed both the number of faults and software metrics to demonstrate the reliability prediction through a case study. Although there method could be applicable to OSS developments and agile developments, metrics measurement results in addition to data of faults are needed; often it is hard to obtain those additional measurement results.

Aman proposed a multistage model that divides the whole development period of OSS into multiple stages, and applies a different growth curve to a different stage[28]. Although its concept is related to our method involving separation of development time periods, target types of data for growth model applications are quite different; its target is code change events while our target is number of issues (faults). We have a plan to compare our method with the multistage model against same issue data.

There is an ongoing challenge to monitor bug-fixing process after releases in OSSs[29]. Our future work could include a investigation of relationship between the bug-fixing process during development and the process after releases in OSSs.

V. Conclusion

Using GSRM, we successfully predict the release dates of OSS. Additionally, we propose a method to evaluate the prediction accuracy, which confirms that GSRM can precisely predict the release date. We believe that our method should be applicable to agile developments because of their iterative and incremental character.

This paper is limited to time-independent development situations in order to compare the two models because NHPP cannot handle time-dependent variables. In the future, we plan to adjust the time-dependence of the models, which may allow GSRM to more accurately predict the number of issues detected. Moreover, we plan to apply our method to agile development projects.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper.

References

[1] Ryushi Shiohama, Hironori Washizaki, Shin Kuboaki, Kazunori Sakamoto and Yoshiaki Fukazawa, "Estimate of the appropriate iteration length in agile development by conducting simulation," Proceedings of the Agile 2012 Conference, pp.41-50, August 13-17, 2012.

[2] Ken Power, "Metrics for Understanding Flow," Proceedings of the Agile 2014 Conference, 2014.

[3] Shigeru Yamada, Mitsuru Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection," IEEE Transactions on Reliability, Vol.32, No.5, 1983.

[4] Kiyoshi Honda, Hironori Washizaki, Yoshiaki Fukazawa, "A Generalized Software Reliability Model Considering Uncertainty and Dynamics n Development," Proceedings of the 14th International Conference of Product Focused Software Development and Process Improvement, pp.342-346, 2013.

[5] Kiyoshi Honda, Hironori Washizaki, Yoshiaki Fukazawa, "Predicting the Release Time Based on a Generalized Software Reliability Model (GSRM)," Proceedings of the 38th Annual IEEE International Computers, Software, and Applications Conference (COMPSAC 2014), pp.604-605, 2014.

[6] Kiyoshi Honda, et al., "Predicting Time Range of Development Based on Generalized Software Reliability Model," Proceedings of the 21st Asia-Pacific Software Engineering Conference (APSEC 2014), pp.351-358, 2014.

[7] Tadashi Dohi, Keiichi Yasui, Shunji Osaki, "Software reliability assessment models based on cumulative bernoulli trial processes," Mathematical and Computer Modelling, Vol. 38, No. 11-13, pp.1177-1184, 2003.

[8] Yoshinobu Tamura and Shigeru Yamada, "A flexible stochastic differential equation model in distributed development environment," European Journal of Operational Research, Vol. 168, No. 1, pp.143-152, 2006.

[9] Yamada Shigeru, Kimura Mitsuhiro, Tanaka Hiroaki, Osaki Shunji, "Software reliability measurement and assessment with stochastic differential equations," IEICE Transactions on undamentals of electronics, communications and computer sciences, Vol. E77-A, No. 1, pp.109-116, 1994.

[10] Xia Cai and Michael R. Lyu, "Software reliability modeling with test coverage: Experimentation and measurement with a fault-tolerant software project," Proceedings of the 18th IEEE International Symposium on Software Reliability Engineering (ISSRE 2007), pp.17-26, 2007.

[11] Yasutaka Kamei, Akito Monden, and Ken-ichi Matsumoto, "Empirical Evaluation of Svm-Based Software Reliability Model," Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2006), Vol. 2, pp.39-41, 2006.

[12] Norm Schneidewind, "A Complexity Reliability Model," Proceedings of the 20th International Symposium on Software Reliability Engineering (ISSRE 2009), pp.1-10, 2009.

[13] Elisabeth A. Nguyen, Carlos F. Rexach, David P. Thorpe, Andrew E. Walther, "The Importance of Data Quality in Software Reliability Modeling," Proceedings of the IEEE 21st International Symposium on Software Reliability Engineering (ISSRE 2010), pp.220-228, 2010.

[14] Hiroyuki Okamura, Yusuke Etani and Tadashi Dohi, "A multi-factor software reliability model based on logistic," Proceedings of the IEEE 21st International Symposium on Software Reliability Engineering (ISSRE 2010), pp.31-40, 2010.

[15] K. Worwa, "A discrete-time software reliability-growth model and its application for predicting the number of errors encountered during program testing," Control and Cybernetics, Vol. 34, No. 2, 2005.

[16] Mohd. Anjum, Md. Asraful Haque, Nesar Ahmad, "Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value", International Journal of Information Technology and Computer Science, Vol. 5, No. 2, pp.1-14, 2013.

[17] Shinji Inoue and Shigeru Yamada, "Lognormal Process Software Reliability Modeling with Testing-Effort," Journal of Software Engineering and Applications, Vol. 6, No. 4, pp.8-14, 2013.

[18] Ron Goldman and Richard P. Gabriel, "Innovation Happens Elsewhere: Open Source as Business Strategy," Morgan Kaufmann, 2005.

[19] Juhani Warsta and Pekka Abrahamsson, "Is Open Source Software Development Essentially an Agile," Proceedings of the 3rd Workshop on Open Source Software Engineering, 2003.

[20] Vinay Tiwari, "Some Observations on Open Source Software Development on Software Engineering Perspectives," International Journal of Computer Science & Information Technology, Vol. 2, No. 6, 2010.

[21] `http://foundation.zurb.com/`

[22] Al Goel, "Software Reliability Models: Assumptions, Limitations, and Applicability," IEEE Trans. Software Engineering, Vol. 11, No. 12, 1985.

[23] Richard Lai and Mohit Garg, "A Detailed Study of NHPP Software Reliability Models," Journal of Software, Vol. 7, No. 6, 2012.

[24] `https://github.com/zurb/foundation`

[25] Chengnian Sun, David Lo, Xiaoyin Wang, Jing Jiang, Siau-Cheng Khoo, "A discriminative model approach for accurate duplicate bug report retrieval," Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE'10), pp.45-54, 2010.

[26] Shigeru Yamada, Hiroshi Ohtera and Mitsuru Ohba, "Testing-domain

dependent software reliability models," Computers & Mathematics with Applications, Vol. 24, No. 1-2, 1992.

[27] Toshiya Fujii, Tadashi Dohi and Takaji Fujiwara, "Towards quantitative software reliability assessment in incremental development processes," Proceedings of the 33rd ACM/IEEE International Conference on Software Engineering (ICSE 2011), pp.41-50, 2011.

[28] Hirohisa Aman, Akiko Yamashita, Takashi Sasaki, Minoru Kawahara, "Multistage Growth Model for Code Change Events in Open Source Software Development: An Example Using Development of Nagios," Proceedings of the 40th EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA 2014), pp.207-212, 2014.

[29] Keisuke Fujino, Akinori Ihara, Kiyoshi Honda, Hironori Washizaki, Kenichi Matsumoto, "Toward Monitoring Bugs-fixing Process after the Releases in Open Source Software," 6th International Workshop on Empirical Software Engineering in Practice (IWESEP 2014), Poster, 2014.