Predicting the Release Time Based on a Generalized Software Reliability Model (GSRM)

Kiyoshi Honda
Waseda University
Tokyo, Japan
Email: khonda@ruri.waseda.jp

Hironori Washizaki
Waseda University
Tokyo, Japan
Email: washizaki@waseda.jp

Yoshiaki Fukazawa
Waseda University
Tokyo, Japan
Email: fukazawa@waseda.jp

***Abstract*—Development environments have changed drastically, development periods are shorter than ever and the number of team members has increased. Especially in open source software(OSS), a large number of developers contribute to OSS. OSS have difficulties in predicting or deciding when it will release. In order to assess recent software developments, we proposed a generalized software reliability model (GSRM) based on a stochastic process, and compared GSRM with other models. In this paper, we focus on the release dates of OSS and the growth of faults(issues).**

I. Introduction

Software reliability is a critical component of computer system availability. Software reliability growth models can be used as an indication of whether enough faults have been removed to release the software. The logistic curve and Gompertz curve[4] are well-known software reliability growth curves. However, these curves cannot account for the dynamics of software development. Developments are affected by various elements of the development environment, such as the skills of the development team and changing requirements.

Here, we propose a model (GSRM) that can describe several development situations that involve random factors, such as the skills of teams and development environments, to provide a time range in which the development will end. Earlier studies only use linear stochastic differential equations; however, our research indicates that non-linear stochastic differential equations lead to more elaborate equations that can model situations more realistically. Moreover, we aim to reveal the development of open source software(OSS).

II. Proposal method

We focus on the determination when OSS release in the point of view of the growth of issues. Especially we use two methods which are below.

A. Separating development terms into each versions.
B. Using GSRM and predicting the number of issues and release dates.

A. Separating terms

The upper side graph of figure 1 indicate the growth of issues about "foundation,"[5] which is a front-end framework, divided by each versions. The shapes of curves are significantly sharped when the new version released. Therefore we scope the changing points of versions and separate them into each versions, and apply our model(GSRM). The reason for separating them is to approximate GSRM more precisely and treat more naturally. and predict the release dates and the numbers of issues.

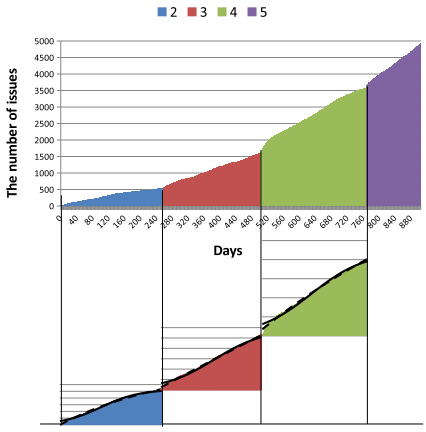

Fig. 1. The number of issues and development days about "foundation."

B. Generalized Software Reliability Model

For our software reliability model, we extend a nonlinear differential equation that describes fault content as a logistic curve to an Ito type stochastic differential equation. We start with the following equation, which is called the logistic differential equation.

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

$N(t)$ is the number of detected faults by time t, a defines the growth rate, and b is the carrying capacity. If $b = 0$, then the solutions of this equation become exponential functions. We extend equation (1) to 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, and express them using a. The time-dependence of a can be used to describe situations such as skill improvements of development members and increases of 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 a focus on the test phase by simulating the number of detected faults. We assume software development to have the following properties.

[0]In our paper[2] we merely given proposal only generalized software reliability model (GSRM) and applied to empirical software. On the other hand, in this paper we target open source software and apply it.

TABLE I. THIS 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 detected faults per unit time is constant, and the uncertainty increase near the end. This model is similar to a logistic curve. (**Model 1-1**)	The number of detected faults per unit time is constant, and the uncertainty is constant at any given time. (**Model 1-2**)	The number of detected faults per unit time is constant, and 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 detected faults 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 detected faults per unit time changes at t_1, and the uncertainty is constant at any given time. (**Model 2-2**)	The number of detected faults per unit time changes at t_1, and 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, and the uncertainty is constant at any given time. (**Model 3-2**)	The number of detected faults per unit time increases, and the uncertainty is greater at the start of project than at the end. (**Model 3-3**)

1) The total number of faults is constant.
2) The number of faults that can be found varies depending on time.
3) 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.

$$dN(t) = (\alpha(t) + \sigma^2/2 + \beta N(t))N(t)dt + N(t)\sigma dw(t) \quad (2)$$

$\alpha(t) + \sigma^2/2 + \sigma dw(t)$ 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 non-linear carrying capacity term. This equation has two significant terms, α and dw; α affects the end point of development, and dw affects the growth curve through uncertainties. In particular, the stochastic term is dependent on $N(t)$, which means that uncertainties depend on the number of detected faults. We compare 3 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$N(t)$). We summarize the types of $\alpha(t)$ and of the coefficient of $dw(t)$ and the corresponding situations in Table I.

Summarizing the above, we can apply the reliability growth models to nine types of development situations. Existing models can describe only one of these situations with additional limitations, but GSRM can describe several of these situations. This is primarily because existing models cannot handle time-dependent growth rates without any limitations, while GSRM can handle the time-dependence, and only the appropriate type of situation needs to be selected as input.

III. APPLICATION TO OSS

We discuss the differences between GSRM and the NHPP models using actual development data of OSS named as "foundation"[5] in a given situation as the growth rate is time-independent. The reason for this limitation is because the NHPP model cannot be applied to other time-dependent situations. We compare GSRM with a general NHPP model on data sets obtained from a github site[6]. In Table II, we show the issues and the days each versions and the residual sum of squares(RSS) and the Akaike's Information Criterion (AIC) about models. This results show GSRM is better than NHPP in the view point of predictions. Predicted numbers of issues and days by GSRM are more precisely than those of NHPP.

TABLE II. COMPARISON OF GSRM WITH OTHER MODELS.

		Actual Data	NHPP	GSRM
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

IV. RELATED WORKS

The software reliability models had ever been used on water fall development, however Fujii et al. developed a quantitative software reliability assessment method in incremental development processes, which is one of agile software developments, based on the familiar non-homogeneous Poisson processes.[1] Fujii et al. did not use only the number of faults but also the software metrics and showed software reliability prediction through a case study.

V. CONCLUSION

Using GSRM, we were able to successfully predict the release dates and the number of issues about OSS. However NHPP can more precisely approximate the growth of issues than GSRM. For future work, we will adjust the time-dependence of models. In this paper, for comparing GSRM with NHPP and the lack of data we limited time-dependence of development, thus GSRM could not more precisely approximate the growth of issues.

REFERENCES

[1] T. Fujii, et al. Towards quantitative software reliability assessment in incremental development processes. ICSE '11, 2011.

[2] K. Honda, et al. A generalized software reliability model considering uncertainty and dynamics in development. PROFES '13 2013.

[3] C. Stringfellow et al. An empirical method for selecting software reliability growth models. *ISSRE '07*, 2007.

[4] S. Yamada, et al. S-shaped reliability growth modeling for software error detection. *Reliability, IEEE Transactions on* 1983.

[5] http://foundation.zurb.com/

[6] https://github.com/zurb/foundation