Predicting Time Range of Development Based on Generalized Software Reliability Model

Kiyoshi Honda, Hidenori Nakai
Hironori Washizaki, Yoshiaki Fukazawa
Waseda University, Tokyo, Japan
Email: khonda@ruri.waseda.jp, hide-and-seek@toki.waseda.jp
{washizaki, fukazawa}@waseda.jp

Ken Asoh, Kaz Takahashi, Kentarou Ogawa
Maki Mori, Takashi Hino, Yosuke HAYAKAWA
Yasuyuki Tanaka, Shinichi Yamada, Daisuke Miyazaki
Yahoo Japan Corporation, Tokyo, Japan
Email: {keasoh, katakaha, keogawa, makimori, tashino
yhayakaw, yasutana, shyamada, daimiyaz}@yahoo-corp.jp

***Abstract*—Development environments have changed drastically, development periods are shorter than ever and the number of team members has increased. These changes have led to difficulties in controlling the development activities and predicting when the development will end. Especially, quality managers try to control software reliability and project managers try to estimate the end of development for planing developing term and distribute the manpower to other developments. In order to assess recent software developments, we propose a generalized software reliability model (GSRM) based on a stochastic process, and simulate developments that include uncertainties and dynamics. We also compare our simulation results to those of other software reliability models. Using the values of uncertainties and dynamics obtained from GSRM, we can evaluate the developments in a quantitative manner. Additionally, we use equations to define the uncertainty regarding the time required to complete a development, and predict whether or not a development will be completed on time. We compare GSRM with an existing model using two old actual datasets and one new actual dataset which we collected, and show that the approximation curve generated by GSRM is about 12% more precise than that generated by the existing model. Furthermore, GSRM can narrow down the predicted time range in which a development will end to less than 40% of that obtained by the existing model.**

I. Introduction

Software reliability is a critical component of computer system availability. Especially, quality managers try to control software reliability and project managers try to estimate the end of development for planing developing term and distribute the manpower to other developments. In a estimating method, a manager collects faults data, which contains the number of faults and the time when faults are detected, and estimates how many faults remained using the faults data applying a prediction model, which is called software reliability growth model. Thus, software reliability growth models have been developed to indicate whether enough faults have been removed to release the software. Although the logistic curve and Gompertz curve [19] are well-known software reliability growth curves, they cannot account for the dynamics of software development, which are affected by various development environment elements, such as the skills of the development team, changing requirements, etc. 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.

Examples of software reliability models include the "Times Between Failures Models" and "Failure Count Models" [6]. We have used the "Failure Count Model," which is based on counting failures and probability methods. The Goel-Okumoto NHPP Model and the Musa Execution Time Model are examples of this type of model [6]. Recent studies by Tamura [16], Yamada [18], Zhang [22], Cai [3], Kamei [9], Dohi [4], Schneidewind [14], Nguyen [11], and Okamura [12] have attempted to describe the dynamics of developments using stochastic differential equations. Although many models have been proposed, surveyed, and compared [17] [2] [10], most failure count models cannot account for the dynamics of development, (e.g., drastic changes in the development team composition or significant reductions in the development time), and cannot precisely predict when developments will end.

Many current models only give the number of faults that will be found within some time range. Here, we propose a model called the generalized software reliability model (GSRM), which can describe several development situations that include random factors (e.g., skills of the development team and development environment), to provide a time range in which development will end. Although earlier studies use linear stochastic differential equations, our research indicates that non-linear stochastic differential equations lead to more elaborate equations that can model situations more realistically. Furthermore, GSRM can quantify uncertainties influenced by random factors. To more accurately predict the time required to complete development and to optimize development teams and environments, uncertainties must be quantified. Thus, this study aims to answer the following three research questions.

- RQ1: How much better is GSRM at describing the growth of software reliability in different situations compared to other models (e.g., NHPP)?
- RQ2: How accurately does GSRM describe the convergence of the number of faults and the appropriate development term in a given situation compared to other models?
- RQ3: How does GSRM predict when a development will end, considering the dispersion due to uncertainty?

Our contributions are as follows:

[0]In our paper[8] we merely given proposal only generalized software reliability model (GSRM). On the other hand, in this paper we suggested the ranges of the term when developments will end and compared GSRM with other model.

- A software reliability model applicable to nine development situations.
- A generalized software reliability model, which is 12% more precise than existing models.
- A new method to predict when development will end.

II. Background

A. Software reliability model

Although many software reliability models have been proposed, the most popular is the non-homogeneous Poisson process (NHPP) model. Hence, we compare GSRM with NHPP using the development data containing the number of faults detected in a given time. Some models quantitatively assess software reliability from the fault data observed in the software-testing phase; similar to these software reliability models, GSRM is based on a fault counting [6] model. The fault counting software reliability model is formulated by counting the number of faults detected in a given time, assuming that faults are detected based on a stochastic process. The NHPP model assumes that the stochastic process governing the relationship between fault detection and a given time interval is a Poisson process. In actual developments, counting the faults predicts the number of remaining faults and provides an indication of when the development will end. To understand how our model was developed, we begin with a description of the general NHPP model. In the NHPP model, the probability of detecting n faults is expressed as

$$Pr\{N(t) = n\} = \frac{\{H(t)\}^n}{n!} \exp\{-H(t)\} \tag{1}$$

$N(t)$ is the number of faults detected by time t, and $H(t)$ is the expected cumulative number of faults detected[13]. Assuming that the number of all the faults is constant at $N_{\max}$, the number of detected faults at a unit time is proportional to the number of remaining faults. These assumptions lead to the following equation

$$\frac{dH(t)}{dt} = c(N_{\max} - H(t)) \tag{2}$$

where c is a proportionality constant. Equation (2) can be solved as

$$H(t) = N_{\max}(1 - \exp(-ct)) \tag{3}$$

Models derived based on equation 3 are called exponential software reliability growth models. The first example of this type of model was proposed by Goel and Okumoto [7]. Later Yamada et al. derived a delayed S-shaped software reliability growth (S-Shaped) model from equation (3) with a fault isolation process [20]. Equation (3) can be rewritten for the delayed S-shaped software reliability growth model using $H_{\mathrm{S}}(t)$, which is the expected cumulative number of faults detected in the S-shaped model, as

$$H_{\mathrm{S}}(t) = N_{max}\{1 - (1 + ct)\exp(-ct)\} \tag{4}$$

B. Motivating example

Existing software reliability growth models give us the number of faults will be found with some ranges of raults, however the models cannot precisely indicate the time when the development will end. Figure 1 shows an example dataset

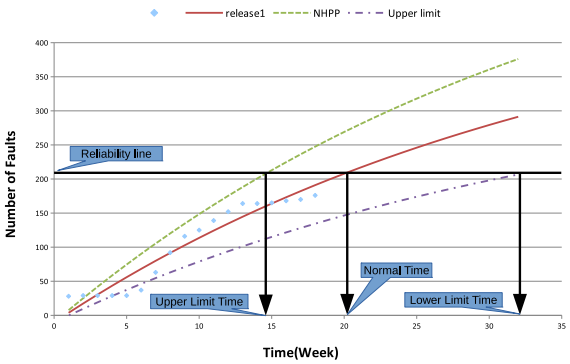

Fig. 1. Time ranges based on NHPP model.

from reference [15] written by C. Stringfellow et al. , which is drawn with three models: the normal NHPP, its upper limit, and its lower limit, which indicate a greater deal of faults than normal NHPP and a less than normal NHPP, whose values are calculated with confidence interval as 70%. For example, the end for the "Lower Limit Time" is twice that of the "Upper Limit Time," indicating that the predictions are not meaningful. Hence, we try to construct a new method which can define the time ranges of development at section 3.2.

In this paper, we compare GSRM with these models. Equation (3) results in an exponentially shaped software reliability graph. However, actual software reliability graphs typically follow a logistic curve or a Gompertz curve [19], which are more complex. Therefore, we propose a new model, GSRM, which can fit either a logistic curve or an exponentially-shaped curve for use in actual developments.

III. Generalized Software Reliability Model

For our software reliability model, we extend a nonlinear differential equation that describes the fault content as a logistic curve into an Ito-type stochastic differential equation. We start with equation. (5), which is called the logistic differential equation.

$$\frac{dN(t)}{dt} = N(t)(a + bN(t)) \tag{5}$$

$N(t)$ 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. Equation (5) can be extended into a stochastic differential equation because actual developments do not strictly obey equation (5) due to the numerous uncertainties and dynamic changes. Such dynamic elements are considered time-dependent and to contain uncertainty; these factors are expressed in a. The time-dependence of a can be used to describe situations such as skill improvement of development members and increases in the growth rate, while the uncertainty of a can describe parameters such as the variability of development members and the environment. We analyze the growth of software with an emphasis on the testing phase by simulating the number of detected faults. Software development is assumed to have the following properties:

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, which can be simulated with Gaussian white noise.

Considering these properties, equation (5) can be extended to an Ito-type stochastic differential equation with $a(t) = \alpha(t) + \sigma dw(t)$ as

$$dN(t) = (\alpha(t) + \frac{\sigma^2}{2} + \beta N(t))N(t)dt + N(t)\sigma dw(t) \quad (6)$$

$N(t)$ is the number of detected faults by time*t*, $\alpha(t)+\sigma^2/2+\sigma dw(t)$ 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 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 $N(t)$. The first type is where $\gamma(t) = N(t)\sigma dw(t)$. The second type is where $\gamma(t)$ does not depend on $N(t)$: $\gamma(t) = \sigma dw(t)$. The third type is where $\gamma(t)$ depends on the inverse of$N(t)$: $\gamma(t) = 1/N(t)\sigma dw(t)$. We vary $\alpha(t)$ and the coefficient of $dw(t)$, and simulate models using equation (6). Table I summarizes the types of $\alpha(t)$ and of the coefficient of $dw(t)$ and their corresponding situations. To apply GSRM, a type in Table I and past data must be selected to calculate the parameters.

A. Uncertainty and Time-dependence

In development, faults are detected and debugged. The detected faults are counted and used to predict when the project will end. A project has a lot of uncertain elements, and the predicted development period is almost never long enough. GSRM can describe the uncertainty of the applied development and calculate the uncertainty of fault detection.

We describe the uncertainty as σdw, which is basically Gaussian white noise and can be obtained from past data. Because the uncertainty is difficult to calculate from equation (5), we assume there are some limits to obtain σdw in quantitative manner. The result can be useful. We start by defining a in terms of σdw from equation (5) as

$$a = \alpha(t) + \sigma dw(t) \quad (7)$$

However, equation (5) cannot be solved due to the time-dependence of a as shown in equation (7). Therefore, we assume that a is time-independent with an added term δ, which is small. This assumption allows equation (5) to be solved, and can be rewritten as

$$\frac{dN(t)}{dt} = N(t)(\alpha + \delta + bN(t)) \quad (8)$$

Equation (8) can be solved as

$$N = \frac{N_{max}}{1 + b\exp\{-(\alpha + \delta)t\}} \quad (9)$$

This equation is a logistic equation where δ is the origin of the uncertainty. $\alpha+\delta$ is the gradient. The sign of δ can be positive or negative. If δ is negative, the gradient of the equation is small, whereas if it is positive, the gradient of the equation is large.

The sign of δ provides the limitation of the uncertainty. If δ is negative (positive), the growth of the graph provides the lower (upper) limit. The lower and upper limits are calculated in the next section. δ is calculated as

$$\delta_i = -\frac{1}{t_i}\ln\left\{\frac{1}{b}\left(\frac{N(t_i)}{N_i} - 1\right)\right\} - \alpha \quad (10)$$

The subscript i indicates that the data is for the ith fault detected at t_i. i differs from the approximate value at t_i. Finally, the average and variance of δ are obtained, which are used to construct an equation for the software reliability growth model.

By using δ and its distribution, which are Gaussian white noise, we can predict the range of the required development period. The range due to uncertainties is obtained using the equations below.

We assume that the detected faults obey equation (8) and that the detection rate has an uncertainty δ that is time independent, which leads to

$$N_+(t) = \frac{N_{max}}{1 + b\exp\{-(\alpha + \delta)t\}} \quad (11)$$

$$N_-(t) = \frac{N_{max}}{1 + b\exp\{-(\alpha - \delta)t\}} \quad (12)$$

For $N_+(t)$, the rate of development is faster than for $N(t)$. For $N_-(t)$, the rate of development is slower than for $N(t)$. Using these equations, we can establish the range from the shortest development period to the longest. The development periods are expressed as

$$t_\pm(t) = -\frac{1}{\alpha \pm \delta}\left[\ln\left\{\frac{1}{b}\left(\frac{N_{max}}{N} - 1\right)\right\}\right] \quad (13)$$

B. Time Range of Development

The development period usually ends when a certain percentage of expected faults (typically 95%) are detected and removed. Using equation (13), the range of the development period can be calculated before the development period actually ends. The range is defined as $\Delta t = t_- - t_+$, and is expressed as

$$\Delta t = \left(\frac{-2\delta}{\alpha^2 - \delta^2}\right)\left[\ln\left\{\frac{1}{b}\left(\frac{N_{max}}{N} - 1\right)\right\}\right] \quad (14)$$

By calculating the development period range in the development, the delay risk can be predicted as well as the delay range. Figure 2 depicts the relations among these equations.

Figure 2 shows the time range of the same data with Figure 1. In figure 1, the time range is 26.30 weeks, however in figure 2 the time range is 6.40 weeks. These results show our method is more meaningful to predict the end of the development than that of NHPP.

TABLE I. THIS COMBINATIONS OF DYNAMICS AS CHARACTERIZED BY $\alpha(t)$ 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**)

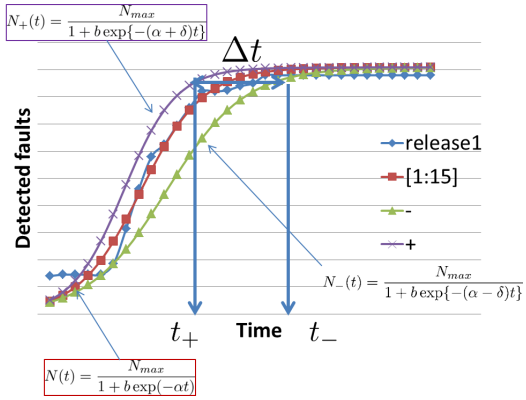

Fig. 2. Relationship between the equations for $N(t)$, $N_+(t)$ and $N_-(t)$, and Δt, t_+ and t_-.

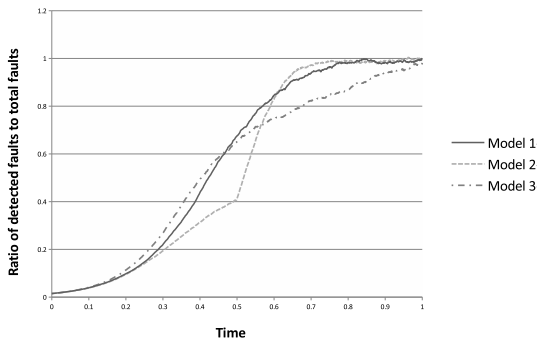

Fig. 3. Plot of the ratio of the cumulative number of detected faults at time t to the total number of detected faults for the entire project where the x-axis represents time in arbitrary units and 1 corresponds to $t_{\max}$ and 0.5 to t_1.

IV. SIMULATION AND DISCUSSION

A. Simulation

The nine cases tabulated in Table I are modeled and plotted by column in Fig. 3, Fig. 4 and Fig. 5. For each column in Table I, the difference between each model is the parameter $\alpha(t)$. In **Model 1-1**, **Model 2-1** and **Model 3-1**, based on **Model 1-1**, we defined $a_2 = a_1$, $a_3 = 2a_1$ and $t_1 = t_{max}/2$ in **Model 2-1**, and $\alpha_3(t) = a_1 t$ in **Model 3-1**. $\alpha(t)$s set in the same manner along all columns (i.e., $\alpha(t)$ is the same along each row in Table I).

For **Model 1-1**, **Model 2-1** and **Model 3-1**, the uncertainty $\gamma(t) = N(t)\sigma dw(t)$, which means that over time, the effect of the uncertainty increases. The situation corresponding to **Model 2-1** is that at time t_1 the number of members of the development team doubles. The situation corresponding to **Model 3-1** is that the members' skills improve over time, effectively doubling the manpower by the time t_{max}. For **Model 1-2**, **Model 2-2** and **Model 3-2**, the uncertainty $\gamma(t) = \sigma dw(t)$, which means that over time, the effect of the uncertainty decreases. For **M**odel 1-3, **Model 2-3** and **Model 3-3**, the uncertainty $\gamma(t) = 1/N(t)\sigma dw(t)$, which means that over time, the effect of the uncertainty decreases.

The purpose of the simulations is to confirm that our approach can assess software reliability under dynamic changes and uncertainties in development, and that it can adapt to the models above and produce appropriate results. We use a Monte Carlo method to examine these models.

1) Model 1-1: The number of detected faults per unit time is constant, and the effect of uncertainty increases over time. As we predicted, the simulation result for **Model 1-1** fits the logistic curve. This result cannot be obtained by using other stochastic models that do not include a non-linear term.

2) Model 2-1: The number of detected faults per unit time changes at t_1, and the effect of uncertainty increases over time. In agreement with our predictions, the resulting curve sharply rises at t_1 and then converges quickly. Other models cannot describe such a time-dependent curve involving a nonlinear term.

3) Model 3-1: Both the number of detected faults per unit time and the effect of uncertainty increase over time. We expected the resulting curve to show a steeper increase than **Model 1-1**, but that was not the case. The reason for this is that the non-linear term pulls the curve down because of the increasing growth rate.

4) Model 1-2: The number of detected faults per unit time is constant, and the effect of uncertainty is constant. As we predicted, the simulation result for **Model 1-2** fits the logistic curve and the uncertainty effects, in dependent of the number of faults. As can be seen in Fig. 4, the curve fits the logistic curve better than for **Model 1-1**. This result cannot be obtained by using other stochastic models that do not include a non-linear term.

5) Model 2-2: The number of detected faults per unit time changes at t_1, and the effect of uncertainty is constant. In agreement with our predictions, the resulting curve sharply rises at t_1 and then converges quickly. Other models cannot describe such a time-dependent curve involving a non-linear term.

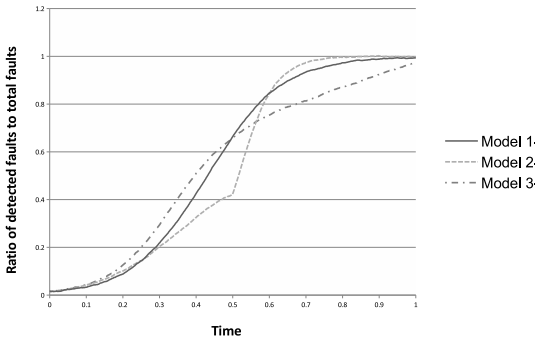

Fig. 4. The same parameters are plotted as in Fig.3, but for Models 1-2, 2-2 and 3-2. In **Model 1-2**, the number of detected faults per unit time is constant. In **Model 2-2**, the number of detected faults per unit time changes at t_1. In **Model 3-2**, the number of detected faults per unit time increases.

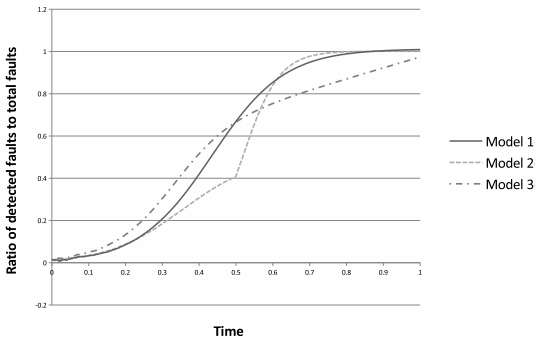

Fig. 5. The same parameters are plotted as in Fig.3, but for Models 1-3, 2-3 and 3-3. In **Model 1-3**, the number of detected faults per unit time is constant. In **Model 2-3**, the number of detected faults per unit time changes at t_1. In **Model 3-3**, the number of detected faults per unit time increases.

6) Model 3-2: The number of detected faults per unit time increases over time, and the effect of uncertainty is constant. We expected the resulting curve to show a steeper increase than **Model 1-2**, but that was not the case. The reason for this is that the non-linear term pulls the curve down because of the increasing growth rate.

7) Model 1-3: The number of detected faults per unit time is constant, and the effect of uncertainty decreases over time. As we predicted, the simulation result for **Model 1-3** fits the logistic curve, and the effect of uncertainty is large at the start of development. This result cannot be obtained by using other stochastic models that do not include a non-linear term.

8) Model 2-3: The number of detected faults per unit time changes at t_1, and the effect of uncertainty decreases over time. In agreement with our predictions, the resulting curve sharply rises at t_1 and then converges quickly. Other models cannot describe such a time-dependent curve involving a non-linear term.

9) Model 3-3: The number of detected faults per unit time increases over time, and the effect of uncertainty decreases over time. We expected the resulting curve to show a steeper increase than **Model 1-3**, but that was not the case. The reason for this is that the non-linear term pulls the curve down because of the increasing growth rate.

TABLE II. COMPARISON OF GSRM AND NHPP MODELS USING DATASET 1.

	NHPP	S-Shaped	GSRM
RSS	67.21	35.14	11.2
AIC	106.5	87.62	59.90

B. Comparison with the NHPP models

The NHPP model is one of many proposed reliability models. In this section, we discuss the differences between GSRM and NHPP models using actual development data for a given situation when the growth rate is time-independent. This limitation is imposed because the NHPP model cannot be applied to other time-dependent situations.

1) Dataset 1: This development dataset is from reference [7] written by Goel and Okukmoto. The data are originally from the U.S. Navy Fleet Computer Programming Center, and consist of the errors in software development. Figure 6 plots the results using the NHPP model and GSRM. The parameters

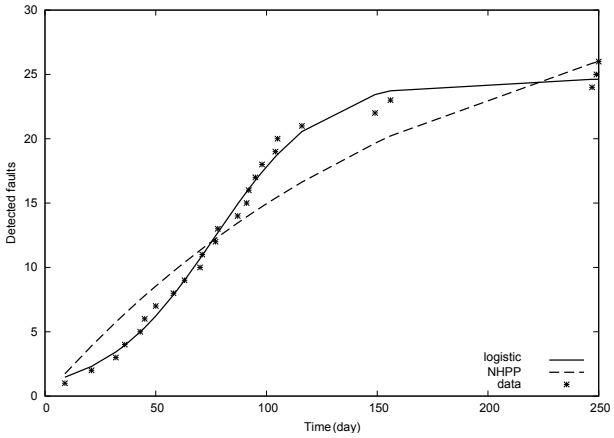

Fig. 6. Comparison of GSRM and the NHPP model.

for both GSRM and the NHPP model are calculated by R [1], which is a language and environment for statistical computing and graphics. The residual sum of squares(RSS) and Akaike's Information Criterion (AIC) are calculated from these models and the development data (Table II). These results show that GSRM provides a better approximation than the NHPP models because GSRM is more flexible due to the non-linear term.

2) Dataset 2: The second development dataset is from reference [15] written by C. Stringfellow et al. The data come from three releases of a large medical record system, which consists of 188 software components (Table III). The data contain the cumulative number of faults and their detected times for the three different releases of a software program. Figures 7-9 plot the results for each release using the NHPP

TABLE III. DATASET 2. NUMBER OF WEEKS FOR DEVELOPMENT AND THE NUMBER OF FAULTS FOR THE THREE DIFFERENT RELEASES OF A LARGE MEDICAL RECORD SYSTEM.

	Weeks	Number of faults
Release 1	18	176
Release 2	17	204
Release 3	13	77

model and GSRM. The parameters are calculated by R [1] for both GSRM and the NHPP model. Then these equations and developmental data are used to calculate RSS and the AIC (Table IV). Furthermore, δ is calculated, and the upper and lower limits are simulated and calculated. Almost all of the real data points are contained within the calculated upper and lower limits. GSRM produces a good fit for release 1 (Figure 7) as the curve for the lower limit corresponds to the worst-case scenario, indicating that if the development is continued until 95% of the 176 faults are detected, five more weeks are necessary than it actually took to complete the development. However, the upper and lower limits are almost the same for the release 2 (Figure 8), suggesting that the development does not have critical uncertainties. Additionally, the GSRM results realize a good fit for release 3 (Figure. 9), and although most of the data points are within the curves for the upper and lower limits, a few are above the upper curve.

TABLE IV. COMPARISON OF GSRM WITH THE NHPP MODELS USING DATASET 2.

		NHPP	S-Shaped	GSRM
Release 1	RSS	4612	3246	1310
	AIC	158.9	152.6	136.3
Release 2	RSS	696.1	3489	473.7
	AIC	119.4	145.8	112.8
Release 3	RSS	264.8	181.1	158.8
	AIC	84.07	79.14	77.43

3) Dataset 3: We collected the third development dataset from Yahoo Japan Corporation in 2013. The data comes from a platform of a search engine. A platform consists of seven major modules: messaging, storage, UI, common, consumer, control-api, and data-api. The modules manage development using Jenkins and track faults with it. Table V shows each module's faults. Figures 10-12 plot the actual number of faults and the predicted number of faults using GSRM for messaging, common and consumer. Table VI compares GSRM to the NHPP model. The results demonstrate that GSRM more accurately.

TABLE V. NUMBER OF FAULTS IN DATASET 3.

Module Name	Days	Number of faults	Predicted faults
messaging	206	240	232.88
storage	194	50	54.63
UI	187	148	144.09
common	184	134	126.72
consumer	157	63	58.50
control-api	190	73	68.08
data-api	183	147	144.60

TABLE VI. COMPARISON OF GSRM AND THE NHPP MODEL USING DATASET 3.

Module Name		NHPP	S-Shaped	GSRM
messaging	RSS	50626	31510	12240
	AIC	1971	1857	1632
storage	RSS	409	592	232
	AIC	253	592	226
UI	RSS	47250	8160	2416
	AIC	1279	1019	841
common	RSS	357852	6824	7124
	AIC	1443	912	920
consumer	RSS	13168	622	514
	AIC	521	329	319
control-api	RSS	1913	1500	784
	AIC	451	433	388
data-api	RSS	9560	3359	1001
	AIC	1036	883	707

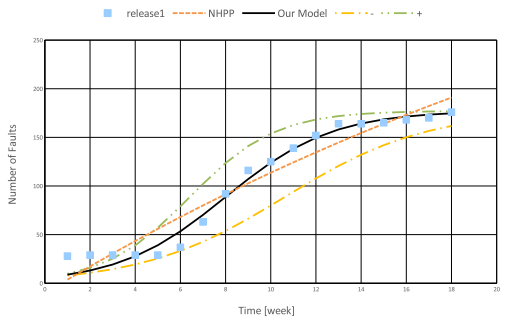

Fig. 7. Cumulative number of detected faults for the entire project of release 1 versus the elapsed number of weeks. release1, NHPP, Our Model, +, and − represent the actual data, the fit using NHPP, the fit using GSRM, the predicted upper limit, and the predicted lower limit, respectively.

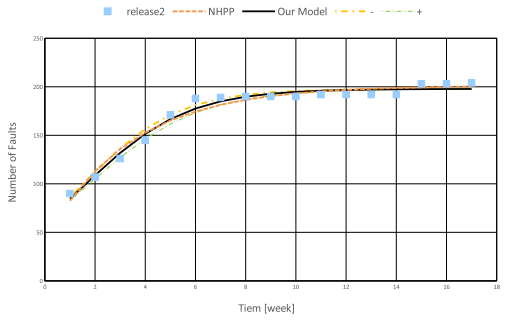

Fig. 8. Cumulative number of detected faults for the entire project of release 2 versus the elapsed number of weeks. Legend is the same as Figure 7.

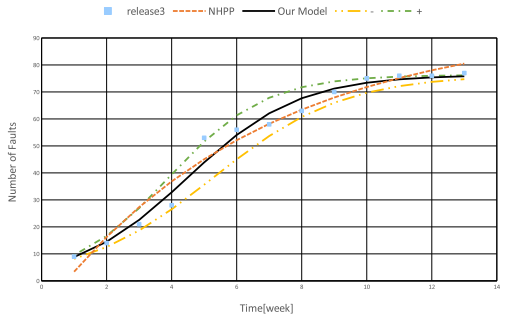

Fig. 9. Cumulative number of detected faults for the entire project of release 3 versus the elapsed number of weeks. Legend is the same as Figure 7.

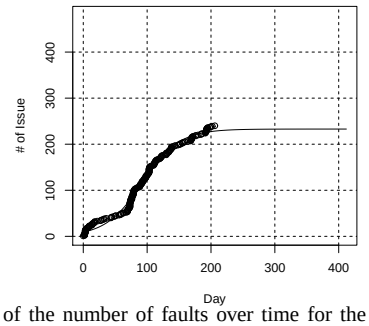

Fig. 10. Plot of the number of faults over time for the messaging module. Circles and solid line indicate the actual faults and predicted faults by GSRM, respectively.

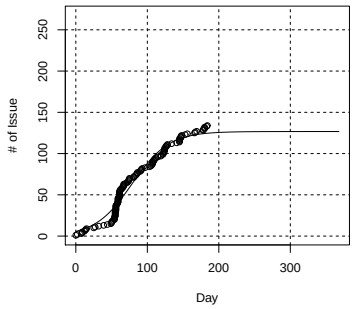

Fig. 11. Plot of the number of faults over time for the common module. Circles and solid line indicate the actual faults and predicted faults by GSRM, respectively.

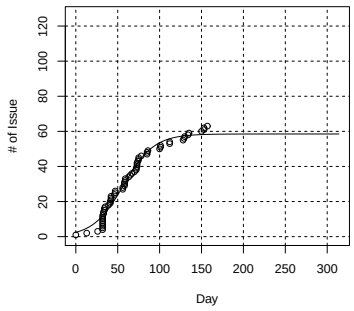

Fig. 12. Plot of the number of faults over time for the consumer module. Circles and solid line indicate the actual faults and predicted faults by GSRM, respectively.

C. Prediction of time ranges

The time range (Δt) when a development is predicted to end is calculated using GSRM as well as by the NHPP model. In the NHPP model, the range is determined by using the upper and lower boundaries to define a confidence interval of 90%.

1) Dataset 2: In dataset 2, Δt is determined by GSRM when the upper and lower boundaries cross the 85% mark of the total number of predicted faults. Table VII shows Δt for GSRM and the NHPP model. Δt's obtained by GSRM have a 40% narrower width than those obtained by the NHPP model, and each dataset lies almost entirely within the time ranges obtained by GSRM. Table VIII shows Δt predicted based on GSRM using a partial dataset. The values of Δt decrease as the development proceeds and the amount of data used for the predictions increases. The results indicate that GSRM can be used to predict delays in development.

TABLE VII. PREDICTED RANGES OF THE NHPP MODEL AND GSRM FOR EACH DATASET. UNIT TIME IS WEEKS.

	NHPP	GSRM
Release 1	31.002	3.739
Release 2	2.890	1.051
Release 3	9.601	3.739

2) Dataset 3: In dataset 3, Δt for GSRM or the NHPP model is determined when the upper and lower boundaries cross the 70% mark of the total number of predicted faults (Table IX).

TABLE VIII. Δt FOR EACH RELEASES. UNIT TIME IS WEEKS.

	All	[1:15]	[1:12]	[1:9]	[1:7]
Release 1	5.314	6.395	8.436	-	-
Release 2	0.771	0.727	0.679	-	0.263
Release 3	2.287	-	4.075	2.754	3.254

TABLE IX. RANGES OF THE NHPP MODEL AND GSRM IN DATASET 3.

Module Name	NHPP	GSRM
messaging	1309	58.13
storage	219	86.92
UI	3308	17.01
common	58162	-
consumer	NaN	-
control-api	421	14.62
data-api	444	72.41

All of the ranges from GRSM are less than those of NHPP. At least one of the NHPP ranges is 2.5 times greater than those of GRSM, demonstrating that the NHPP's range is not meaningful for predicting when development will end.

Table X shows Δt of each module's faults and that predicted using GSRM and a partial dataset. By comparing to the data in Table V, some modules are well predicted during development, and almost all the GSRM results are well fitted and meaningful to predict when development will end. However, the common and consumer modules cannot be predicted because the predictions using partial data are inaccurate. The prediction using the partial dataset for the common (consumer) module is 101.74 (98.03) days, while the actual development is 184 (157) days.

D. Summary

Wide applicability (RQ1): Our simulations applied the reliability growth models to nine types of development situations, which are characterized by two uncertainty elements related to fault detection. Although existing models can describe only one of these situations with additional limitations, GSRM can describe several of these situations. This is primarily because existing models cannot handle time-dependent growth rates without limitations. In contrast, GSRM can handle time-dependence, and only the appropriate type of situation must be inputted. Additionally, GSRM has a scheme for development uncertainties and can construct a model involving uncertainties.

Comparison with NHPP model (RQ2): Given a situation where the growth rate is time-independent, we used two actual datasets and compared GSRM to the NHPP model. The results show a precise convergence of the numbers of faults and the appropriate development terms with GSRM. The convergence precision is at least 12% higher for GSRM than for the NHPP model, demonstrating that GSRM can describe software growth more realistically than previously proposed models based on the NHPP model. Thus, GSRM may provide developers with a more accurate plan for releasing software.

Predictions involving uncertainties (RQ3): For two datasets, GSRM is able to model the uncertainties, and calculate Δt to predict not only the total development time, but also how long development will be delayed due to uncertainties. Δt cannot be obtained with other models. Therefore, existing models can only predict the day when the development will be end, but not the length of a delay.

Internal validity threats: In comparing models, we use two datasets, both of which were obtained by one organization or company. Therefore, the data could contain mistakes or

TABLE X. NUMBER OF FAULTS IN DATASET 3.

Module Name	Predicted end days	Predicted left days	Number of faults	Predicted faults	Predicted range
messaging	173.33	25	198	230	58.13
storage	201.36	60	41	49	86.92
UI	184.69	57	117	173	17.01
common	101.74	-23	107	96	-
consumer	98.03	-3	52	53	-
control-api	199.85	76	58	97	14.62
data-api	156.39	43	124	145	72.41

some other false elements. Moreover, the data were too old to compare with recent developments. However, recent studies also use these data so the validity should be protected.

External validity threats: We only tested GSRM with two datasets, which is insufficient to make generalizations about GSRM. Moreover, the datasets are old and the scales of their systems are smaller than recent systems. A future project will use datasets related to large-scale systems. Additionally, we only compared GSRM with the NHPP model. There are a lot of other proposed and applied models. Although these other models have similar origins as the NHPP model, GSRM should also be compared to them.

V. RELATED WORK

Many different types of software reliability growth models exist. Yamada et al. proposed an extend NHPP model, which is related to test-domain dependence [21]. 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.

Although water fall development has not been applied to software reliability models, Fujii et al. developed a quantitative software reliability assessment method via incremental development processes, which is a type of agile software development based on the familiar non-homogeneous Poisson processes [5]. Fujii et al. used both the number of faults and software metrics to demonstrate software reliability predictions via a case study.

VI. CONCLUSION

Using GSRM, we successfully simulated developments containing uncertainties and dynamic elements. We obtained the time-dependent curve, which is not possible using other models. GSRM can be used to predict the length of the development when the team composition drastically changes during development as well as to simulate and analyze nine types of developments. Furthermore, we were able to define uncertainty values from actual data containing information on the faults during development, and apply GSRM to three datasets to calculate Δt, including the range of possible development times considering the uncertainty values. We also examined how well GSRM can predict the required development time using actual datasets. By using past data, GSRM can calculate the uncertainties with and predict how long a project will take. In the future, we aim to find methods to quantitatively evaluate teams or team members considering uncertainties and to optimize teams to suit particular projects using GSRM.

REFERENCES

[1] The r project for statistical computing. http://www.r-project.org/

[2] M. Anjum *et al*. Analysis and ranking of software reliability models based on weighted criteria value. *I.J. Information Technology and Computer Science*, 2013.

[3] X. Cai *et al*. Software reliability modeling with test coverage: Experimentation and measurement with a fault-tolerant software project. *ISSRE*, 2007.

[4] T. Dohi *et al*. Software reliability assessment models based on cumulative bernoulli trial processes. *Mathematical and Computer Modelling*, 38, 2003.

[5] T. Fujii *et al*. Towards quantitative software reliability assessment in incremental development processes. *ICSE* , 2011.

[6] A. Goel. Software reliability models: Assumptions, limitations, and applicability. *Software Engineering, IEEE Transactions on*, SE-11(12),1985.

[7] A. GOEL *et al*. Time-dependent error-detection rate model for software reliability and other performance measures. *Reliability, IEEE Transactions on*, R-28(3), 1979.

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

[9] Y. Kamei *et al*. Empirical evaluation of an svm-based software reliability model. *ISESE*, 2006.

[10] R. Lai *et al*. A detailed study of nhpp software reliability models. *JOURNAL OF SOFTWARE*, 7(6), 2012.

[11] E. A. Nguyen *et al*. The importance of data quality in software reliability modeling. *ISSRE*, 2010.

[12] H. Okamura *et al*. A multi-factor software reliability model based on logistic regression. *ISSRE*, 2010.

[13] S. Osaki, editor. *Stochastic Reliability and Maintenance Modeling*. Springer Berlin Heidelberg, 2002.

[14] N. Schneidewind *et al*. A complexity reliability model. *ISSRE*, 2009.

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

[16] Y. Tamura *et al*. A flexible stochastic differential equation model in distributed development environment. *European Journal of Operational Research*, 168(1), 2006.

[17] 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*, 34, 2005.

[18] S. Yamada *et al*. Software reliability measurement and assessment with stochastic differential equations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, E77-A(1), 1994.

[19] S. Yamada *et al*. S-shaped reliability growth modeling for software error detection. *Reliability, IEEE Transactions on*, R-32(5), 1983.

[20] S. Yamada *et al*. s-shaped software reliability growth models and their applications. *Reliability, IEEE Transactions on*, R-33(4), 1984.

[21] S. Yamada *et al*. Testing-domain dependent software reliability models. *Computers and Mathematics with Applications*, 24(12), 1992.

[22] N. Zhang *et al*. Software reliability measurement and assessment with stochastic differential equations. In *World Automation Congress 2012*, 2012.