Defect Analysis and Prediction by Applying the Multistage Software Reliability Growth Model

Jieming Chi*, Kiyoshi Honda*, Hironori Washizaki*, Yoshiaki Fukazawa*
Kazuki Munakata†, Sumie Morita†, Tadahiro Uehara†, and Rieko Yamamoto†
*Waseda University, 3-4-1 Ohkubo, Shijuku-ku Tokyo, JAPAN
Email: tko@fuji.waseda.jp, khonda@ruri.waseda.jp, {washizaki, fukazawa}@waseda.jp
† Fujitsu Labs Ltd., 4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa 211-8588, JAPAN
Email: {munakata.kazuki, morita.sumie, uehara.tadahiro, r.yamamoto} @jp.fujitsu.com

***Abstract*—In software development, defects are inevitable. To improve reliability, software reliability growth models are useful to analyze projects. Selecting an expedient model can also help with defect predictions, but the model must be well fitted to all the original data. A particular software reliability growth model may not fit all the data well. To overcome this issue, herein we use multistage modeling to fit defect data. In the multistage model, an evaluation is used to divide the data into several parts. Each part is fitted with its own growth model, and the separate models are recombined. As a case study, projects provided by a Japanese enterprise are analyzed by both traditional software reliability growth models and the multistage model. The multistage model has a better performance for data with a poor fit using a traditional software reliability growth model.**

Keywords—Software Reliability; Growth Model; Multistage Modes.

I. INTRODUCTION

Software reliability is an important part of software as it ensures both quality and stability. Because bugs accompany software during development, it is important to establish a method to check and evaluate software reliability. For companies, the software reliability growth model is a good solution because it can be used as an indicator of the number of potential failures [1]. Therefore, in the past few years, more and more companies tried applied software reliability growth models to ensure software quality [2][3][4][5][6].

Developing a model curve for the ratio that bugs occur will increase the efficiency of software development. Applying a growth model to the actual data, which includes bug information, is a useful method. Although numerous models have been developed, a universal model has yet to be established. In particular, some data may be poorly fitted by a single software reliability growth model.

Herein we try to resolve the situation where all the actual data about information does not fit one growth model well. We use the multistage model as a new modeling method to fit defect information, improving the fit when some of the data is ill fitted using one software reliability growth model.

Specifically, we address the following questions:

RQ1: Is there a metric to evaluate the fit of a model?

RQ2: Is the multistage model better suited for software reliability? What metric can be used to evaluate the suitability of the model?

Both researchers and members of the company involved in the case study have a vested interested in this study. As researchers, we hope to find a way to ensure software reliability in software development and release, especially in the case where a traditional software reliability growth model cannot meet the required accuracy. The members of the software development company hope to discover a process that realizes a smoother and more economical development process using a predictive model.

II. BACKGROUND

Ensuring software reliability is critical when developing and releasing a project. A growth model of defects is a common method to check and verify software reliability.

A. Software Reliablity Growth Models

For companies, a product's software reliability is mainly evaluated by focusing on the amount of failures found during the testing periods [7].

Here a software reliability growth model means a single model is used to analyze defects. (This differ from the multistage model.) In theory, there are many basic software reliability growth models. Examples include the Non-homogeneous Poisson process (NHPP, a very popular growth model in various fields), Weibull, etc. [8]. In this research, the logistic model and the Gompertz model are used as single software reliability growth models. Both are representative trend models that produce an S-shaped curve.

The curve to fit data in the logistic model is expressed as

$$y_L(t) = \frac{a}{1 + e^{-b(t-c)}} \quad (1)$$

The logistic model is widely applied in the scientific community [9]. Here $y_L(t)$ means the number of failures that are detected by *t*. When *t* is sufficiently long, the value of $y_L(t)$ is equal to *a*.

On the other hand, the data fitted by the Gompertz model is given as

$$y_G(t) = ae^{-bct} \qquad (2)$$

The Gompertz model has been applied to various science and technology fields to model biological phenomena, economic phenomena, etc. [10]. Similar to the logistic model, $y_G(t)$ indicates the number of failures detected by *t*, and the result also equals *a* if *t* is big enough.

We need a tool to calculate these parameters. Here we use R Programme to calculate them. R is a programming language and environment for statistical computing and graphics [11]. All the parameters and the information of curve can be calculated by R Programme.

B. Multistage Model

The multistage model does not use only one-rule or obey a formula. Instead, it selects several ideal models and combines them into one model. In previous works, the multistage model has been used to solve problems about poor-fitting models [8][15]. Although not all previous works employ the multistage model as a solution for defect analysis, the data in these projects are satisfied with stochastic event conditions. This model can work well when a traditional software reliability growth model fits poorly. Section IV describes the multistage model for software reliability.

III. Traditional Software Reliability Growth Models

A software reliability growth model is a component of the multistage model. The multistage model is a combination of several traditional software reliability growth models. The application of traditional models and an evaluation method can also be used in multistage models. Section IV compares the prediction capability of the multistage model to a traditional software reliability growth model.

A. Proposal to detect the appropriate software reliability growth model for each stage of the multistage model

The following steps are used to detect unexpected situations or defects in a software reliability growth model.

(1) Acquire and normalize the data source from the company.
(2) Use R to apply a software reliability growth model to the actual data.
(3) Detect and analyze the results.

Although important, normalizing the data is time-consuming. In this study, data from the company is ordered chronologically. The data indicates only the time and type of failure. After the normalization, the data is arranged as a list of dates, which record how many bugs occur on a particular day.

The result after fitting can be used for the prediction. Here we applied the software reliability growth model to Project 1. In a well-fit model curve, the predicted amount of defects is similar to the model's parameter *a*.

In the analysis, we need to evaluate which model better fits the actual data. If the software reliability growth model cannot fit the actual data perfectly, then the model is not applicable for predictions. Hence, the model needs to be altered or another model must be used to analyze the project.

B. Results of software reliability growth models

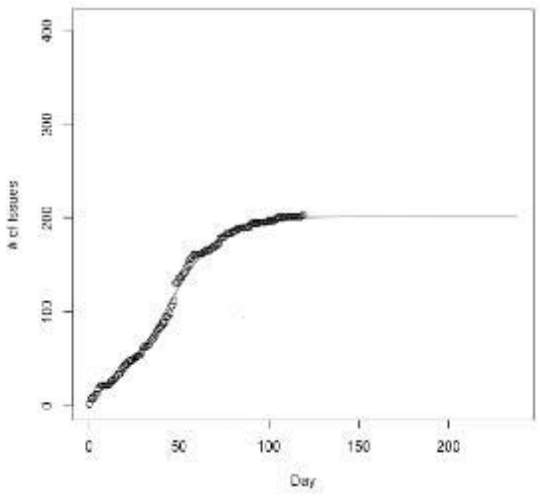

Fig. 1. Plot of Project 1. Model curve is fitted by the logistic model. Plot shows the actual data.

We applied both the logistic model (Fig. 1) and the Gompertz model (Fig. 2) to the data of Project 1. The curves indicate that the data is reasonably well fitted by the models. However, a visual evaluation of the models is insufficient to determine which model is better at detecting more than 95% of the defects. Consequently, our analysis focuses on two questions. (1) Which model is the first (in days) to observe more than 95% of the defects? (2) Which model is better for this project? We used the residual sum of squares (RSS) as an evaluation tool. Table 1 shows the results of the RSS ratio and the date that 95% defects are found for each model.

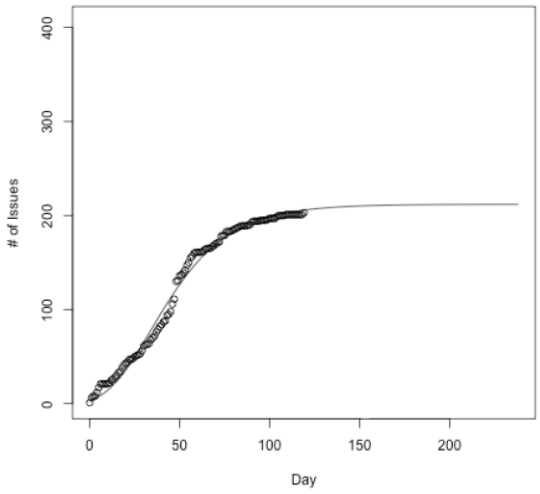

Fig. 2. Plot of Project 1. Model curve is fitted by the Gompertz model. Plot shows the actual data.

The RSS ratio of the logistic model is lower than that of the Gompertz model, indicating that the logistic curve is a better fit for the data in Project 1 In this project, 95% of all defects are found around the 86th day. These results imply that the company can fix this project in 86 days when the model is well fitted to the actual data. However, it is possible that the model does not fit the data. In this case, the prediction using the model is meaningless. Therefore, the logistic model is to compared the multistage model for Project 1.

TABLE I. RSS RATIO AND DATE OF 95% DEFECTS LIST BETWEEN TWO MODELS OF PROJECT 1.

	logistic	Gompertz
RSS Ratio	0.69	1.00
Calculated Date	85.67	105.57

IV. MULTISTAGE MODEL

The multistage model uses more than one model. It divides the data into stages. Then the best software reliability growth model is applied to each stage. Next all the models are combined to connect the stages. In the multistage model, the new model is implemented when the previous stage ends.

To verify the most suitable model for each stage, each stage must be evaluated before building the multistage model. In this research, we analyzed Project 2, which is not well fitted using a traditional software reliability growth model (Fig. 3).

A. Building and evaluating the multistage model

In this research, the multistage model contains two models: the logistic model and the Gompertz model. Because the multistage model is flexible and scalable, it can be extended to include three or more models. However, it may encounter a problem. Not all of the models have three parameters like the logistic or the Gompertz model. For example, the Weibull model can also be used for software reliability growth modeling [12], but it contains four parameters. If RSS is used as an evaluation tool to compare the logistic and Weibull model, the RSS value for the Weibull model will larger than that of the logistic model. Therefore, we used the Akaike information criterion (AIC) to evaluate the stage models [13].

The AIC value increases as the date becomes larger. Here $f(t)$ is used to present the daily normalized AIC value, which is given by [14]

$$f(t) = \frac{\mathrm{AIC}(t)}{t} \qquad (3)$$

Here t means the date. AIC(t) means the AIC value evaluated from the first day until this date.

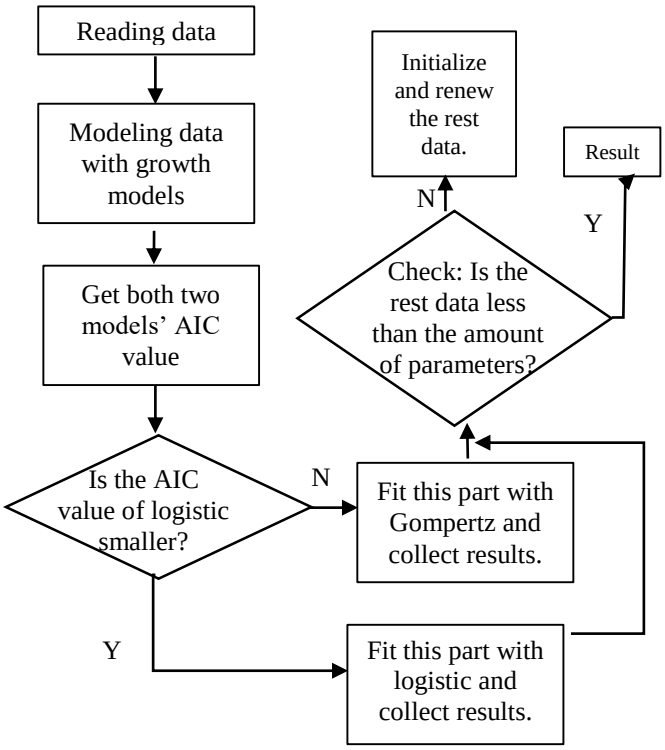

Fig.4 Flowchart of the process to create a multistage model

To determine the best-fitted stage part, we need to find the minimum value of *f(t)* because this indicates the case with the best reliability. First, we applied two software reliability growth models to the data. Then the *f(t)* value by day is listed for two models. We assumed that *t'* is the date with the minimum value, and we reduced the data to this period as a stage model. Then we

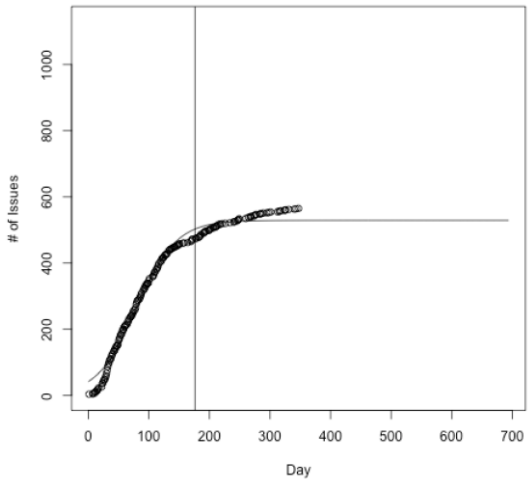

Fig. 3. Plot picture of Project 2, fitted by logistic model. The actual data have a poor-fit with the convergence, and it RSS value is far higher than normal.

noted the corresponding growth model for this stage. Afterwards, the date period from 0 to *t'* is used. Next we set the (*t'*+1) date as the start, and determined the second stage model. We repeated this process until the last stage model is identified. Figure 4 shows a flowchart of the process.

B. Result and discussion of the multistage model

Table 2 lists the stages and Fig. 5 shows the multistage model. The red plot and black curve denote the actual data and the optimized multistage model, respectively. The colorized vertical lines indicate the gaps between stages.

We can infer that in the early periods, the project system is unstable due to the many defects. The gaps are intensive and concentrated in the early part, which means a great volatility in this project system. After long-term maintenance, the improvement in the defects slows, and eventually the number of bugs converge.

The 7th stage has a long duration in the Gompertz model. The previous software reliability predictions started in the middle of a release [15]. This method is quite suitable for the multistage model. Here we use the 7th stage for analysis and the prediction.

Table 3 compares the results to the traditional software reliability growth models. Our multistage model has a lower AIC value, implying a higher reliability. The convergence number of defects is 543. Thus, 95% of the defects are found on the 205th day. The multistage model seems to have better prediction abilities compared to the logistic and the Gompertz models. (The Gompertz model has a higher AIC value than the logistic model.)

TABLE II. AIC RATIO AND DATE OF THE 95% DEFECTS LIST OF PROJECT 2.

	logistic	Multistage
AIC Ratio	1.00	0.92
Calculated Date	175.95	205.10

TABLE III. STAGE INFORMATION OF PROJECT 2 IN THE MULTISTAGE MODEL.

No.	Period of Date	Duration	Fitted Model
1	1 – 22	22	logistic
2	23 – 36	14	Gompertz
3	37 – 39	3	Gompertz
4	40 – 43	4	Gompertz
5	44 – 46	3	Gompertz
6	47 – 67	21	Gompertz
7	68 – 347	280	Gompertz

V. CONCLUSION

Both the software reliability growth model and the multistage model are successfully fitted to the data of two projects. Moreover, we developed an evaluation and prediction tool, allowing the company in the case study to appropriate allocate engineers for the debugging process, conserving resources and costs. For RQ1, we used the RSS value and the AIC value as standards to evaluate the growth models. For RQ2, AIC is used, demonstrating that the multistage model can provide a good solution when a project is ill fitted with a traditional growth model.

Although the applicability of the multistage model is confirmed using a case study of a company, it is applicable to other database defects with stochastic events. This method provides a new point of view in software reliability growth models.

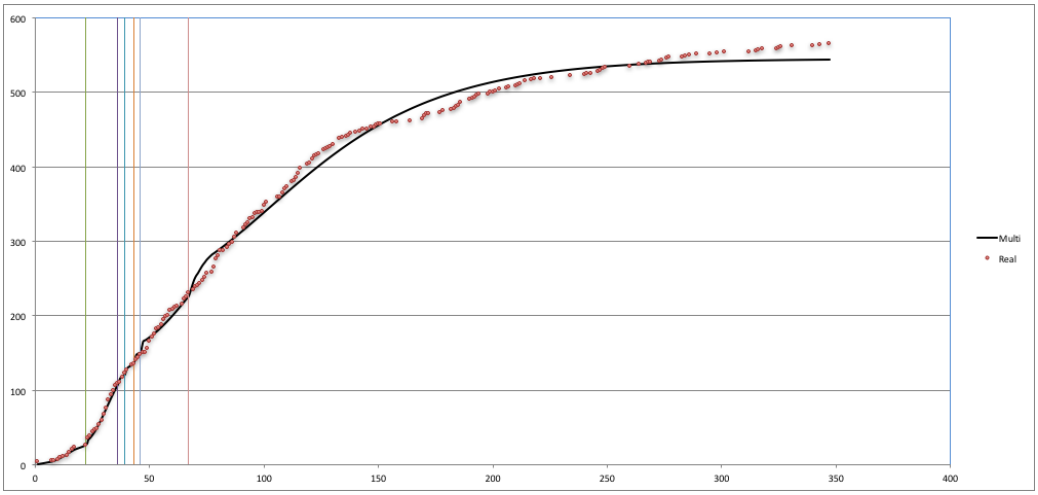

Fig. 5. Final Multistage model.

In the future, we plan to expand the multistage model to consider other growth models (e.g., the average combination of different growth models). These new models should contribute to software reliability and may help resolve problems that are difficult to solve using traditional software reliability growth models.

REFERENCES

[1] A. Wood, "Software reliablity growth models," *Technical Report* 96.1, 1996, pp. 1.

[2] A. Goel, "Software reliability models: Assumptions, limitations, and applicability," *Software Engineering, IEEE Transactions on*, vol. SE11, no. 12, pp. 1411–1423, Dec 1985.

[3] S. Yamada, M. Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection," *Reliability, IEEE Transactions on*, vol. R-32, no. 5, pp. 475–484, Dec 1983.

[4] S. Yamada, M. Kimura, H. Tanaka, and S. Osaki, "Software reliability measurement and assessment with stochastic differential equations," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 77, no. 1, pp. 109–116, 1994.

[5] S. Yamada, "Recent developments in software reliability modeling and its applications," in *Stochastic Reliability and Maintenance Modeling*. Springer, 2013, pp. 251–284.

[6] X. Cai and M. Lyu, "Software reliability modeling with test coverage: Experimentation and measurement with a fault-tolerant software project," in *Software Reliability, 2007. ISSRE '07. The 18th IEEE International Symposium on*, Nov 2007, pp. 17–26.

[7] A. Wood, "Predicting software reliablity," *Computer*, vol. 29, no. 11, pp. 69-77, 1996.

[8] H. Aman, A. Yamashita, T. Sasaki and M. Kawahara, "Multistage growth model for code change events in open source software development: an example using development of Nagios," in *Euromicro Conference on Software Engineering and Advanced Applications (2014 IEEE)*, 2014, pp. 207-212.

[9] A. Tsoularis and J. Wallace, "Analysis of logistic growth models," *Mathematical Biosciences*, vol. 179, no. 1, pp. 21-55, 2002.

[10] C. P. Winsor, "The Gompertz curve as a growth curve," in *Proc. National Academy of Sc.*, vol. 18, no. 1, 1932, pp 1-8.

[11] K. Honda, H. Washizaki, Y. Fukazawa, K. Munakta, S. Morita, T. Uehara and R. Yamamoto, "Detection of unexpected situations by applying software reliability growth models to test phases," in *2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW 2015)*, 2015, pp 2-5.

[12] S. Yamada, J. Hishitani, and S. Osaki, "Software reliability growth with a Weibull test-effort: a model and application," *IEEE Trans. Reliability*, vol. 42, no. 1, pp. 100-106, 1993.

[13] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automatic Control*, vol. 19, no. 6, pp. 716-723, 1974.

[14] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *2nd International Symposium on Information Theory*, Petrov, B. N., and Caski, F. (eds.), Budapest: Akadimiai Kiado, pp. 267–281, 1973.

[15] K. Okumoto, "Customer-perceived software reliability predictions: beyond defect prediction models," *Stochastic Reliability and Maintenance Modeling*, vol. 9, no. 11, pp. 219-249, 2013.