Case Study: Project Management Using Cross Project Software Reliability Growth Model Considering System Scale

Kiyoshi Honda*, Nobuhiro Nakamura†, Hironori Washizaki* and Yoshiaki Fukazawa*
*Waseda University, 3-4-1 Ohkubo, Shijuku-ku Tokyo, Japan
Email: khonda@ruri.waseda.jp, {washizaki, fukazawa}@waseda.jp
† Sumitomo Electric Industries, Ltd., 4-5-33, Kitahama, Chuo-ku, Osaka, Japan
Email: nakamura-nobuhiro@sei.co.jp

***Abstract*—We propose a method to compare software products developed by the same company in the same domain. Our method, which measures the time series of the number of detected faults, employs software reliability growth models (SRGMs). SRGMs describe the relations between faults and the time necessary to detect them. Herein our method is extended to classify past projects for comparison to current projects to help managers and developers decide when to end the test phases or release a project. Past projects are classified by three parameters: lines of code, number of test cases, and test density. Then SRGM is applied. Our extended method is applied to the datasets for nine projects developed by Sumitomo Electric Industries, Ltd. Classification by test density produces the best results.**

I. Introduction

Several researchers have proposed software reliability growth models (SRGMs), which have been used to assess and predict software reliability. These models, which are applied to one project dataset, predict the number of faults that will be detected. Prior to developing such a model, several faults must be identified. In industrial studies, managers often want to predict the number of faults in a current project based on previous projects in the same domain and scale. However, previous models do not always predict the new project. Moreover, if a project does not have the same domain and scale as a past project, a previous model cannot be applied. In such situations, managers and developers cannot determine when to end the test phases or release a project.

We proposed a cross project SRGM to monitor a project by comparing it other past projects [1]. Our method creates a leveled SRGM from old project datasets and helps managers and developers decide when to end the test phases or release a project by comparing the situation of the new project and a leveled SRGM. Since the leveled SRGM contains all kinds of projects, not all projects can be compared with the leveled SRGM. For example, since one project is always under the leveled SRGM, the managers of the project cannot decide to end test phases.

In this paper, we extend our method by classifying the projects contained within the leveled SRGM. Prior to the test phases, we selected system scale parameters such as the LOC, number of test cases, and test density (which is defined as the number of test cases divided by LOC) as classification parameters to create a leveled SRGM.

A. Research Questions

This study aims to answer the following research questions:

1) RQ1: Do the results from the classified leveled SRGMs differ from those of the unclassified SRGMs?
2) RQ2: If the results differ, which classification more precisely describes the results?

Our contributions are as follows:

- Three types of classified SRGMs are compared in nine empirical projects.
- A method to monitor the progress of a project is derived.

In this paper, we classify and compare three leveled SRGMs in nine empirical projects. The results indicate that the leveled SRGMs classified by test density tend to be a good fit. Thus, employing leveled SRGMs classified by test density can help managers and developers determine when to each the test phases or release a project.

II. Background

Because reliability is a crucial component when releasing software, several approaches have been proposed to measure it. Software development includes numerous uncertainties and dynamics regarding the development processes and circumstances.

A. Software Reliability Growth Model (SRGM)

This section treats some example software reliability models, while the next section explains our model. Although many software reliability models have been proposed, the most popular is the non-homogeneous Poisson process (NHPP) model. A recent study has suggested that the Logistic model followed by the Gompertz [2] model is the most suitable with respect to fitness [3]. In this study, we employ the Gompertz model using development data containing the number of faults detected for a given time. These models are common in Japan.

The Gompertz model is given by

$$N_{\mathrm{G}}(t) = N_{\max} \exp(-A_{\mathrm{G}} {B_{\mathrm{G}}}^{t}) \tag{1}$$

where $N_{\mathrm{G}}(t)$ is the number of faults detected by time t. If $t \to \infty$, $N_{\mathrm{G}}(t)$ becomes $N_{\max}$ ($0 < B_{\mathrm{G}} < 1$). The parameters,

$N_{\max}$, A_{G} and B_{G} can be calculated using the Levenberg-Marquardt method with R.

B. Project monitoring

Although multiple methods have been proposed to monitor projects, there are several concerns in software development. The Engineering Project Management using the Engineering Cockpit is one method to manage and monitor project situations [5]. It provides developers and managers with project specific information.

Nakai *et al*. studied how to identify the state and the quality of a project based on goal, question, metric (GQM) method [6] and project monitoring [7]. They employed Jenkins, which is a continuous integration tool to visualize and collect fault data, lines of codes, test coverage, etc. Then they evaluated the project status using the collected data based on the GQM method.

Ohira *et al*. developed the Empirical Project Monitor (EPM), which automatically collects and analyzes data from versioning histories, mail archives, and issue tracking records from multiple software repositories [8]. EPM provides graphs of the collected and analyzed data to help developers and managers. However, EPM is not applicable to analyze SRGMs or to visualize the results.

C. Motivating Example

Figure 1 shows the results of our method, which were obtained by a leveled SRGM from the datasets for nine projects developed by Sumitomo Electric Industries, Ltd. Leveled SRGMs do not seem appropriate for projects P2 and P5 because these projects are far from the leveled SRGM line.

In Figure 1, we show the results of our method. The results are obtained a leveled SRGM from the datasets for nine projects developed by Sumitomo Electric Industries, Ltd. It would seem to be not good for project P2 and P5 to use the leveled SRGM's since project B and E are far from the leveled SRGM's line.

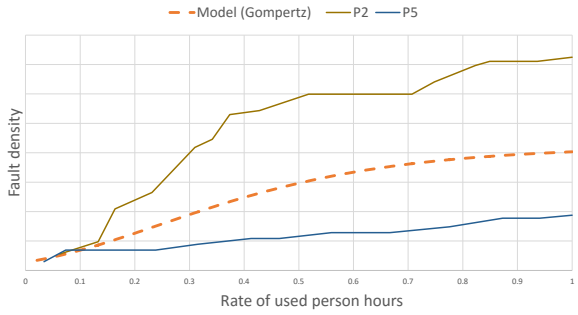

Fig. 1. Fault densities and rates of used person hours for projects P2 and P5 and the leveled Gompertz model

III. Proposal of Classified Leveled SRGM Considering System Scale

We propose that a classified leveled SRGM considering the system scale should resolve the project dependency.

A. Extension of SRGM to fault densities

The equation of the Gompertz model for fault densities and rates of used person hours is given as

$$D_{\mathrm{G}}(t') = D_{\max} \exp(-A'_{\mathrm{G}} {B'_{\mathrm{G}}}^{t'}) \tag{2}$$

where $D_{\mathrm{G}}(t')$ is the number of faults detected by the rate of used person hours t'. If $t' \to \infty$, $D_{\mathrm{G}}(t')$ becomes $D_{\max}$ $(0 < B'_{\mathrm{G}} < 1)$. The parameters, $D_{\max}$, A'_{G} and B'_{G} can be calculated using the Levenberg-Marquardt method with R.

B. Comparison of projects

Figure 2 overviews our method, which compares the results of SRGMs between projects with different lines of code, numbers of test cases, total person hours, and number of faults. Our method has three steps:

1) Divide the number of detected faults by the created lines of code for all data. Convert the person hours to the rate of used person hours.
2) 2) Merge all the data into one dataset. Rearrange the data in chronological order.
3) Apply a SRGM to the new dataset.

We consider the SRGM from the new dataset to be a leveled SRGM of all datasets.

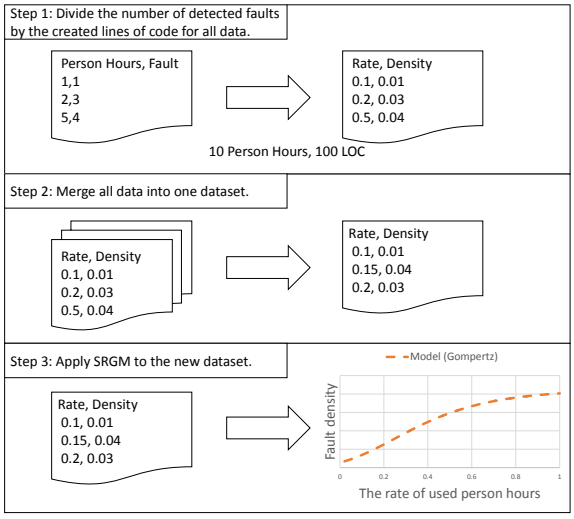

Fig. 2. Overview to compare the results of SRGM between projects.

The first step converts the fault data of each project into the fault density and the rate of used person hours because the number of faults and terms depend on the project. If only the number of faults and person hours are treated, the effort of the developers and project difficulty cannot be evaluated. Additionally, we assume that the fault densities and the rates of used person hours can be used to compare and monitor projects because the fault density values are the same and the rate of used person hours converge.

The second step merges the converted datasets into one dataset to create an averaged SRGM. Moreover, to model the merged dataset, the data is rearranged in chronological order.

This study models the dataset to an SRGM by a nonlinear least-squares method through R.

The third step applies the merged dataset to the SRGM based on the fault densities and the rate of used person hours. The results indicate the leveled line of development, which can be used to help managers and developers assess the progress of a development. Deviation of the dataset from the leveled line means that a development is not going well at a given time.

IV. Evaluation and Results

We evaluated our method via case studies. Then we applied our proposed method to the datasets from nine projects developed by Sumitomo Electric Industries, Ltd. using the same framework. It is should be noted that figures and tables do not indicate actual values because the information is confidential.

A. Evaluation design and result

To answer RQ1 (Do the results from the classified leveled SRGMs differ from those of the unclassified SRGMs?) and RQ2 (If the results differ, which classification more precisely describes the results?), we compared the differences between models classified by lines of code (LOC), the number of estimated test cases (test case), and test density. Specifically, we applied the Gompertz model to nine project datasets and classified them into two groups by the median of each value. Table I shows the details of projects. Then we calculated the residual sums of square (RSS) for each model and compared the results. RSS indicates the differences between the actual data and a model, where a small value indicates a good model fit.

TABLE I
Details of projects.

Project	LOC	Number of test case	Test density
P1	Small	Large	Large
P2	Small	Small	Large
P3	Large	Large	Small
P4	Small	Small	Large
P5	Large	Small	Small
P6	Large	Small	Small
P7	Small	Small	Small
P8	Large	Large	Small
P9	Small	Large	Large

In this evaluation, we collected data from nine projects from Sumitomo Electric Industries, Ltd., including lines of code, number of fault, number of estimated test cases, and the time series of detected fault in days and person hours. We compared the unclassified SRGM (Figure 3) to the SRGMs classified by LOC (Figure 4), test case (Figure 5), and test density (Figure 6). In Figures Figure 3 – 6, the x-axis represents the rate of used person hours, while the y-axis indicates the fault density. The legends, which are the same in Figs. 3 – 6, denote the nine project datasets, which are labeled P1 to P9.

B. Discussion

1) RQ1 (Do the results from the classified leveled SRGMs differ from those of the unclassified SRGMs?): Table II shows

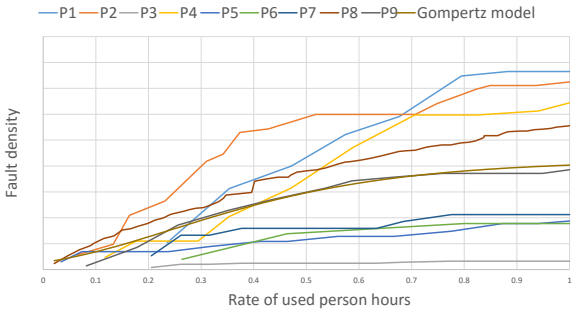

Fig. 3. Results of the unclassified SRGM and the projects.

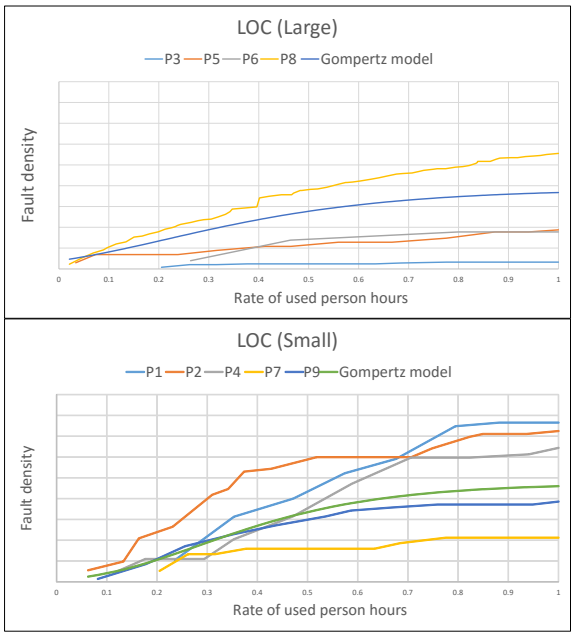

Fig. 4. Results of the SRGM model classified by LOC and the projects.

the RSS of the classified and unclassified leveled SRGMs. Each value indicates the RSS of the model. The sum is the total of the values of the large group and the values of the small group. The results of the SRGM do differ based on the classification.

TABLE II
Comparison of the RSS of the classified and unclassified leveled SRGMs.

Classification	Large	Small	Sum
None	-	-	161.80
Case	97.15	52.56	149.71
LOC	96.29	59.74	156.03
Density	19.15	104.7	123.85

2) RQ2 (If the results differ, which classification more precisely describes the results?): Table II indicates that the most precise model in the large group is the classification by test density, but this is the worst model in the small group. However, for the total optimization, the classification by test density gives the most precise model. In the large and the small

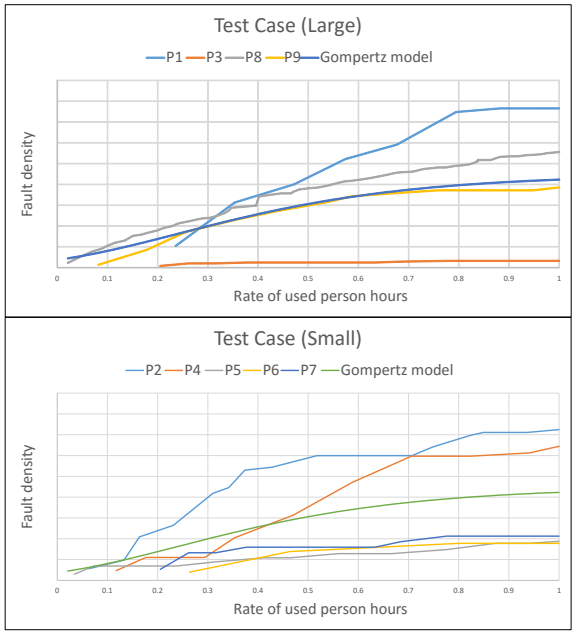

Fig. 5. Results of the SRGM model classified by the test case and the projects.

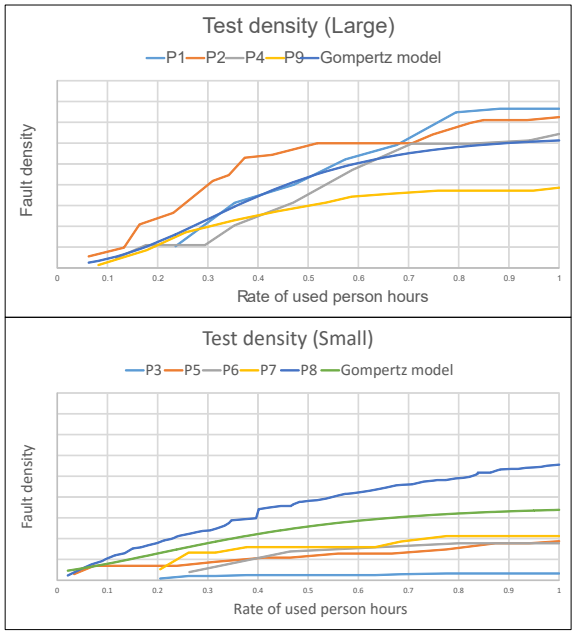

Fig. 6. Results of the SRGM model classified by the test density and the projects.

groups, the classification by LOC and test case yield almost the same value. In the total optimization, the unclassified SRGM provides the worst model.

Figure 7 shows the Gompertz model classified by the test density and P2, and P5. The leveled SRGMs more precisely describe the data than the unclassified leveled SRGM in Figure 1. Thus, the leveled SRGM classified by the test density has the smallest RSS in these models, implying that the classification by test density gives the most precise model.

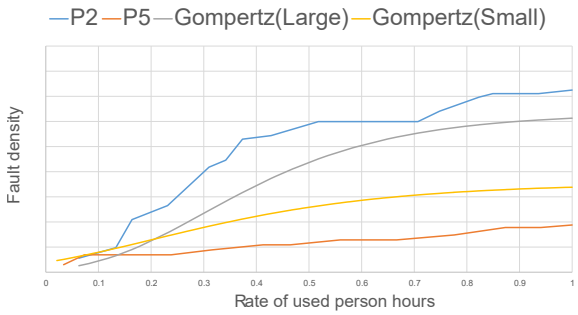

Fig. 7. Fault densities and rates of used person hours for P2 and P5 and the leveled Gompertz models classified by test density.

V. Conclusion

We proposed a leveled SRGM which treated cross project datasets by classifying system scales of projects to compare software products developed by the same company in the same domain. We successfully modeled nine actual datasets by classifying with system scale parameters. The SRGM classified by test density can more precisely model the data than other classifications, including no classification.

In the future, we plan to use other dividing methods such as the k-means clustering since this work divided nine projects into two group by the median.

Acknowledgment

This work has been conducted as a part of “Research Initiative on Advanced Software Engineering in 2015” supported by Software Reliability Enhancement Center (SEC), Information Technology Promotion Agency Japan (IPA).

References

[1] K. Honda, N. Nakamura, H. Washizaki, and Y. Fukazawa, “Case study: Project management using cross project software reliability growth model (to appear in),” in *International Workshop on on Trustworthy Computing In conjunction with QRS 2016*, Aug 2016.

[2] K. Ohishi, H. Okamura, and T. Dohi, “Gompertz software reliability model: Estimation algorithm and empirical validation,” *Journal of Systems and Software*, vol. 82, no. 3, pp. 535 – 543, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121208002604

[3] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Torner, “Evaluating long-term predictive power of standard reliability growth models on automotive systems,” in *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*, Nov 2013, pp. 228–237.

[4] “The r project for statistical computing,” `http://www.r-project.org/`.

[5] T. Moser, R. Mordinyi, D. Winkler, and S. Biffl, “Engineering project management using the engineering cockpit: A collaboration platform for project managers and engineers,” in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, July 2011, pp. 579–584.

[6] V. R. Basili, G. Caldiera, and H. D. Rombach, “The goal question metric approach,” in *Encyclopedia of Software Engineering*. Wiley, 1994.

[7] H. Nakai, K. Honda, H. Washizaki, Y. Fukazawa, K. Asoh, K. Takahashi, K. Ogawa, M. Mori, T. Hino, Y. Hayakawa, Y. Tanaka, S. Yamada, and D. Miyazaki, “Initial industrial experience of gqm-based product-focused project monitoring with trend patterns,” in *Software Engineering Conference (APSEC), 2014 21st Asia-Pacific*, vol. 2, Dec 2014, pp. 43–46.

[8] M. Ohira, R. Yokomori, M. Sakai, K.-i. Matsumoto, K. Inoue, and K. Torii, “Empirical project monitor: A tool for mining multiple project data,” in *International Workshop on Mining Software Repositories (MSR2004)*. IET, 2004, pp. 42–46.