Case Study: Project Management Using Cross Project Software Reliability Growth Model

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. Although several researchers have studied cross project defect predictions to determine defect locations using the features of previous software product's code such as lines of codes and complexities, past works on SRGMs did not compare products or develop comparison methods. Herein we propose a method to compare SRGMs across products. To provide managers and developers insight on advances of its products, our method is applied to the datasets for nine projects developed by Sumitomo Electric Industries, Ltd. SRGMs based on person hours are between 13% and 97% more precise than those based on calendar time.**

***Index Terms*—Software Reliability Growth Model, Faults prediction, Empirical study.**

I. Introduction

Over the past few decades, several companies have employed software reliability growth models (SRGMs) to evaluate reliability, which is an important component of software [1] [2] [3] [4] [5]. However, SRGMs have several issues. SRGMs sometimes misfit the actual data in ongoing developments. In addition, the results do not always match the developers' expectations.

If SRGMs misfit tha actual data, the managers and developers will decide wrong plans, for example stopping testing early or release software which has not been tested enough. On the other hand, if the results of SRGMs indicate that the faults are enough detected or not enough detected contrary to the managers and developers' expectations, they will decide wrong plans too. If software is released with several faults left, the company which have released it will take time to debug it or cause damage or affect negatively to its users.

In order to avoid such misfitting and mismatching the developers' expectations, we propose a new good accuracy SRGMs using person hours. We assume that SRGMs based on calendar time cannot realize accurate predictions, because many kinds of SRGMs treat calendar time, which includes holidays and non-testing time and does not reflect the actual efforts by developers.

A. Research Questions

This study aims to answer the following research questions:

1) RQ1: Do the results from SRGMs based on person hours differ from those based on calendar time?
2) RQ2: If the results differ, which model more precisely describes the relation between faults and detection time?
3) RQ3: Are there any metrics that can compare the progress against other developments?
4) RQ4: If such metrics exist, can they compare the progress between different developments?

Our contributions are as follows:

- SRGMs based on person hours and calendar time are compared in nine empirical projects.
- A method to compare SRGMs is derived.
- A method to monitor the progress of a project in person hours is developed and implemented in test cases.

In this paper, we compared the SRGMs based on person hours and calender time in nine empirical projects. The results indicated the SRGMs based on person hours tend to be good fitting, so using SRGMs based on person hours would make precise plan.

II. Background

Several approaches have been proposed to measure reliability due to its importance when releasing software. Some approaches model fault growth, which is a type of SRGM. Software development includes numerous uncertainties and dynamics regarding development processes and circumstances. This section explains SRGMs, their uncertainties and dynamics as well as provides a motivating example.

We have proposed a model called the Generalized Software Reliability Model (GSRM) to treat the uncertainties and dynamics regarding development processes and circumstances[6]. Previously, we have predicted the release times of open source software (OSS) using GSRM [7] and agile development [8]. Additionally, we have applied GSRM to company's datasets [9].

A. Software Reliability Growth Model (SRGM)

This section treats some example software reliability models, while the next section explains our model. Although numerous models have been proposed, the most famous is the non-homogeneous Poisson process (NHPP) model.

Some methods quantitatively assess software reliability from fault data observed in the software-testing phase using

a software reliability model based on the counting faults [1] model. Similarly, our approach is also based on the counting faults model. By counting the faults and measuring the detection time, a software reliability model is formulated assuming that fault detection is based on a stochastic process. The NHPP model assumes that the stochastic process for the relationship between fault detection and detection time is a Poisson process. In actual developments, counting faults predicts the remaining faults and provides an indication about the end of the development.

First consider the general NHPP model, where the probability of detecting n faults is described as

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

where $N(t)$ is the number of faults detected by time t, $H(t)$ is the expected cumulative number of faults detected [10]. Assuming that the total number of faults is constant, N_{max}, the number of detected faults at a unit of time is assumed to be proportional to the remaining faults. These assumptions are expressed as

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

where c is a proportionality constant. The solution to the above equation is

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

This model, which is called an exponential software reliability growth model, was originally proposed by Goel and Okumoto [11]. In this paper, we compare our model to this model.

Equation (3) provides an exponential shaped graph. However, in actual developments the software reliability graph does not fit the exponential shaped; it usually fits a logistic curve or Gompertz curve[12], which are more complex than an exponential shaped graph. Consequently, we propose a new model that can fit a logistic curve or an exponential shaped curve for use in actual developments.

Although many software reliability models have been proposed, the most popular is the non-homogeneous Poisson process (NHPP) model, but a recent study has suggested that the Logistic model followed by the Gompertz model are the most suitable with respect to fitness [13]. In this study, we employ the Logistic model and the Gompertz model using development data containing the number of faults detected for a given time. These models are common in Japan.

The Logistic model is expressed by

$$N_{\mathrm{L}}(t) = \frac{N_{\max}}{1 + \exp\{-A_{\mathrm{L}}(t - B_{\mathrm{L}})\}} \tag{4}$$

where $N_{\mathrm{L}}(t)$ is the number of faults detected by time t. If $t \to \infty$, $N_{\mathrm{L}}(t)$ becomes $N_{\max}$. The parameters, $N_{\max}$, A_{L} and B_{L} can be calculated using R [14], which is a language and environment for statistical computing and graphics.

The Gompertz model is given by

$$N_{\mathrm{G}}(t) = N_{\max} \exp(-A_{\mathrm{G}} {B_{\mathrm{G}}}^t) \tag{5}$$

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 R.

B. Project monitoring

Although multiple methods exist 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 [15]. 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) [16] and project monitoring [17]. 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 [18]. 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

Figures 1 and 2 show our motivating examples. Figure 1 indicates that the number of faults based on calendar time, which includes holidays and non-testing time, does not reflect the actual efforts by developers. We hypothesize that SRGMs based on calendar time cannot realize accurate predictions. Herein the accuracy of a model based on calendar time is compared to that based on person hours by evaluating nine projects developed by Sumitomo Electric Industries, Ltd.

Figure 2 shows an example of two fault datasets, where the x-axis represents the calendar time and the y-axis represents the number of faults. For comparison, the scales of the x- and y-axes are the same for Project B and Project E. Therefore, the total time is longer and more faults are detected for Project B than Project E.

However, criteria to compare these projects do not exist because each project depends on its lines of domain, code, developers, budget, etc. In this paper, we evaluate the fault density as functions of calendar time, person hours, and number of tested cases. This evaluation realizes a method to compare different projects.

III. Proposal to compare SRGM between projects

We propose an extension of SRGM to cover fault densities as well as a method to apply the person hours to SRGMs.

A. Extension of SRGM to fault densities

The equation of the Logistic model for fault densities and rates of used person hours is given by

$$D_{\mathrm{L}}(t') = \frac{D_{\max}}{1 + \exp\{-A'_{\mathrm{L}}(t' - B'_{\mathrm{L}})\}} \tag{6}$$

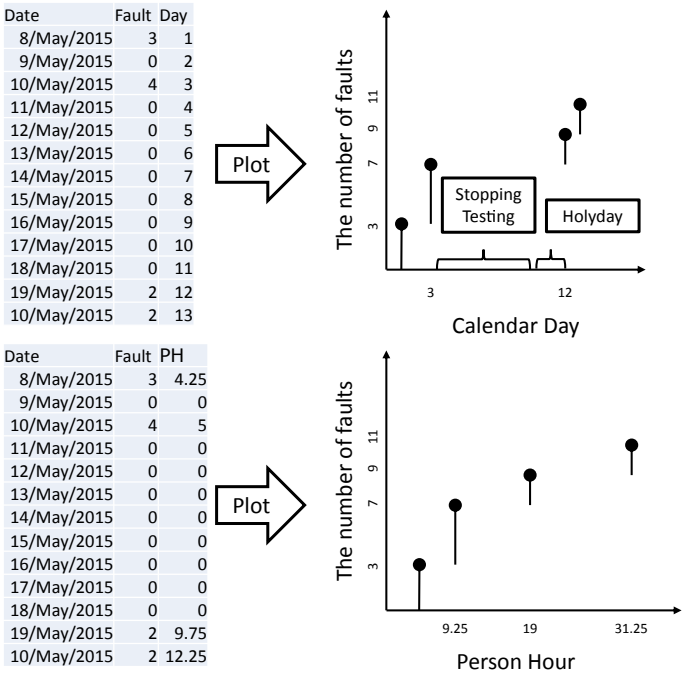

Fig. 1. Examples of calendar time and person hours.

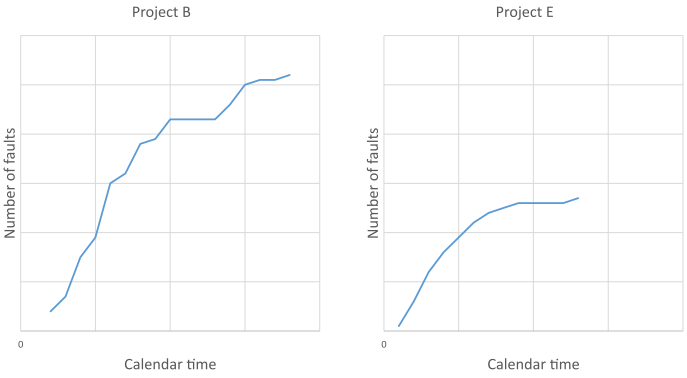

Fig. 2. Example of a comparison between projects.

where $D_{\mathrm{L}}(t)$ is the fault density by the rate of used person hours t'. If $t' \to \infty$, $D_{\mathrm{L}}(t)$ becomes $D_{\max}$. The parameters, $D_{\max}$, A'_{L} and B'_{L} can be calculated using R. The equation of the Gompertz model for fault densities and rates of used person hours is given as

$$D_{\mathrm{G}}(t') = D_{\max} \exp(-A'_{\mathrm{G}} B'^{\,t'}_{\mathrm{G}}) \tag{7}$$

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 R.

B. Comparison of projects

Figure 3 overviews our method, which compares the results of SRGMs between projects with different lines of code, numbers of test cases, total person hours, and numbers 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) Merge all data into one dataset. Rearrange it into chronological order.
3) Apply SRGM to the new dataset.

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

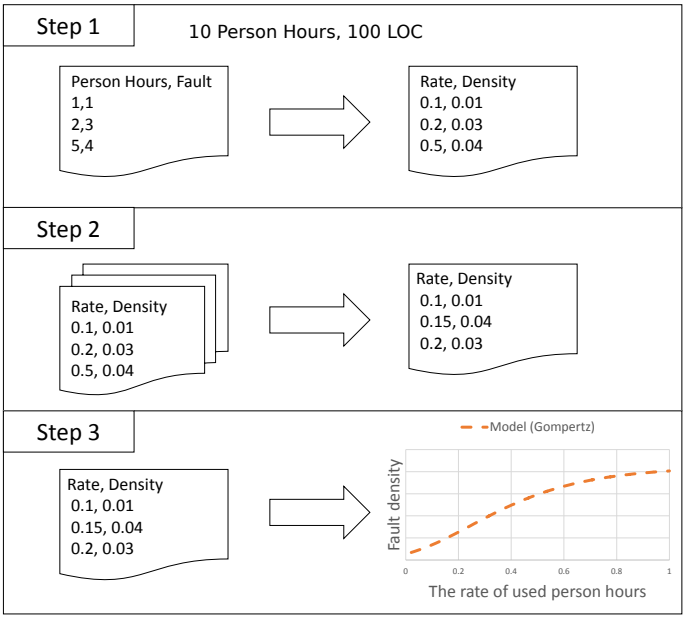

Fig. 3. Overview to compare the results of SRGM between projects.

The first step converts the fault data of each project into the fault density and the rate of used person hours because the numbers of faults and the terms depend on the project. For example, consider a scenario where 100 person hours are required to detect 20 faults in Project 1, but 50 person hours are necessary to detect 10 faults in Project 2. If only the number of faults and person hours are treated, the effort of the developers and the difficulty of project cannot be evaluated. Additionally, we assume that the fault densities and the rates of used person hours are values that can be used to compare and monitor projects because the fault densities values are the same and the rate of used person hours converge.

The second step merges the converted dataset 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 SRGM by a nonlinear least-squares method through R.

The third step applies the merged dataset to 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. If the dataset for a development strays from the leveled line, it means that the development is not going well at that time.

IV. Evaluation and Results

We evaluated our method via case studies. Then we applied our proposed method to datasets from nine projects developed by Sumitomo Electric Industries, Ltd. using the same framework.

A. Evaluation design and result

To answer RQ1 (Do the results from SRGMs based on person hours differ from those based on calendar time?) and RQ2 (If the results differ, which model more precisely describes the relation between faults and detection time?), we compared the differences between models based on calendar time and person hours. Specifically, we applied the Logistic model and the Gompertz model to nine project datasets using calendar time and person hours. Then we calculated the residual sums of square (RSS) for each model and compared the results. RSS indicates the differences between actual data and a model. A small RSS value indicates the model is a good fit for the actual data.

To answer RQ3 (Are there any metrics that can compare the progress against other developments?) and RQ4 (If such metrics exist, can they compare the progress between different developments?), we compared the correlations between the metrics in the collected datasets, the lines of code that only developers created, the numbers of faults, the number of test cases estimated by developers, the number of test cases that developers tested, calendar time, and person hours. Specifically, we evaluated the correlations between the metrics and then applied the Logistic model and the Gompertz models, which are based on the fault density, person hours, and test cases, to the calculated RSS for each model. Finally, we compared the correlation to answer RQ3. Then we compared the RSS results and interviewed the managers to quantitatively and qualitatively answer RQ4.

In this evaluation, we collect nine projects data from Sumitomo Electric Industries, Ltd. including function points, lines of code, number of fault, a number of estimated test cases and the time series of detected fault days, a number of implemented test cases, and the person hours. These projects are developed for business applications throgh web from 2013 to 2015 with a same framework which has been developed by Sumitomo Electric Industries, Ltd.

1) Comparison between calendar time and person hours: We compared SRGMs based on calendar time (Figure 4) to those based on person hours (5). In Figure 4 (5), the x-axis represents the calendar time (person hours) and the y-axis represents the number of faults. In Figure 6, the x-axis represents the number of implemented test cases and the y-axis represents the number of faults. The legends, which are the same in Figs. 4 – 6, indicate the nine project datasets, which are labeled A to I.

Figures 4 and 5 suggest that there is a difference in understanding the growth of faults between calendar time and person hours. To clearly understand this difference, we applied SRGMs (the Logistic model and the Gompertz model) to the calendar datasets and person hour datasets, and evaluated the

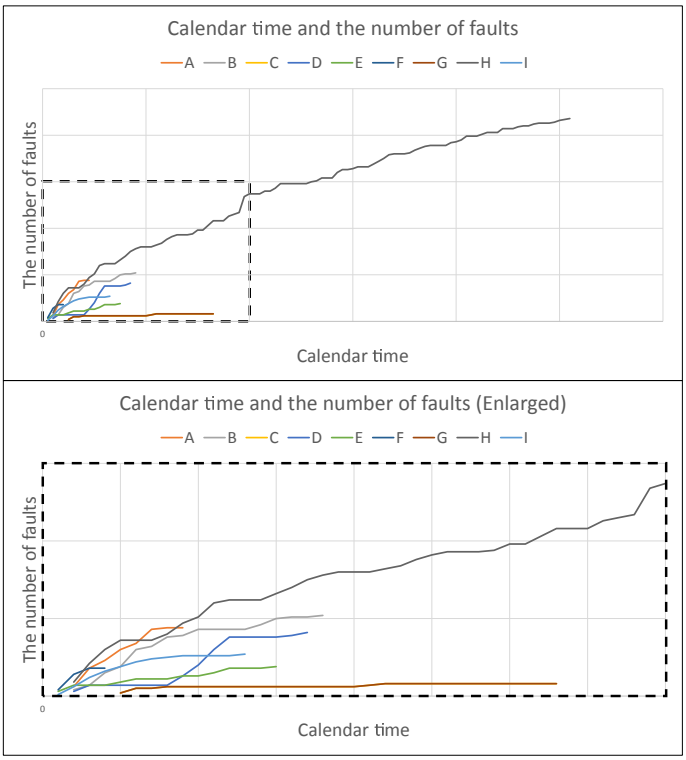

Fig. 4. Relation of the number of faults and calendar time.

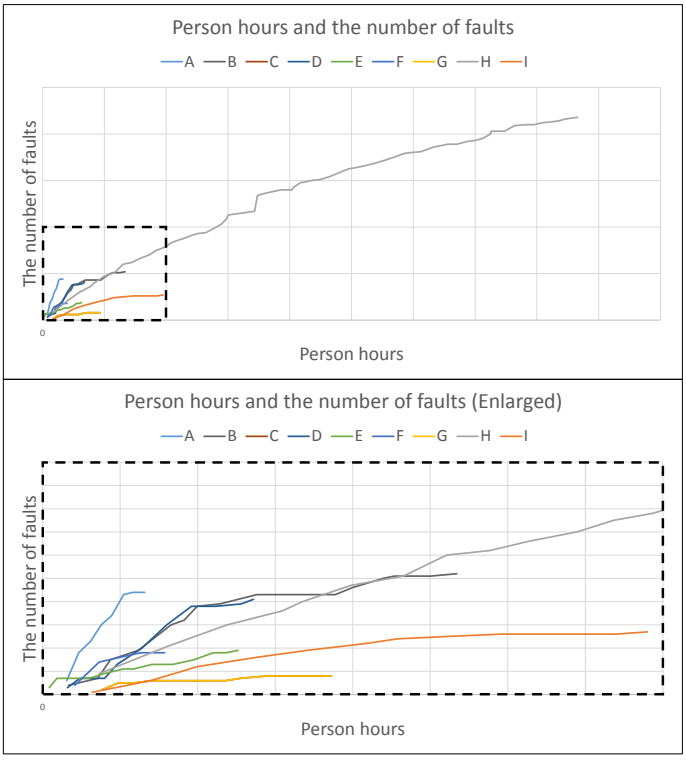

Fig. 5. Relation of the number of faults and person hours.

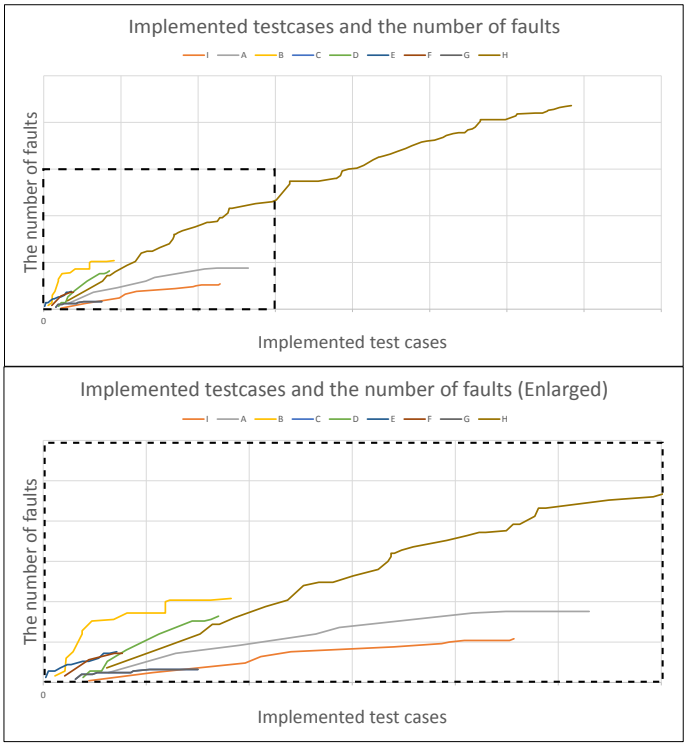

Fig. 6. Relation of the number of faults and implemented test cases.

residual sum of squares (RSS), which is typically used to evaluate the differences between the data and the model (Table I). The model with the best fit has the smallest RSS. For all datasets, SRGMs based on calendar time produce larger RSSs than SRGMs based on person hours.

2) Comparison of values: To evaluate the correlations between each value, we compared the function points (FP), lines of code (LOC), number of fault (Fault), fault densities of LOC (Fault/LOC), fault densities of FP (Fault/FP), number of estimated test cases (TC), number of implemented test cases (TC), total calendar time, and total person hours (Table II). All the values are from the end of the project. Because the datasets are developed within the same framework, the FP represents the function points targeting the extended function. Similar to FP, the developers created the LOC. In Table II, the Fault correlation values of the person hours, Implemented TC, and Calendar time are large (0.95582, 0.93433, and 0.92996, respectively), indicating that the number of faults is strongly related to person hours, number of implemented test cases, and calendar time.

3) Compare the projects: We compared the results of SRGMs based on person hours to SRGMs based on the implemented test cases for the nine projects (Table III).

4) Rates of used person hours: Figure 7 shows the results of the model using fault densities and the rate of used person hours, where the x-axis represents the rate of used person hours and the y-axis represents the fault density. The legend indicates the nine project datasets, which are labeled as A to I.

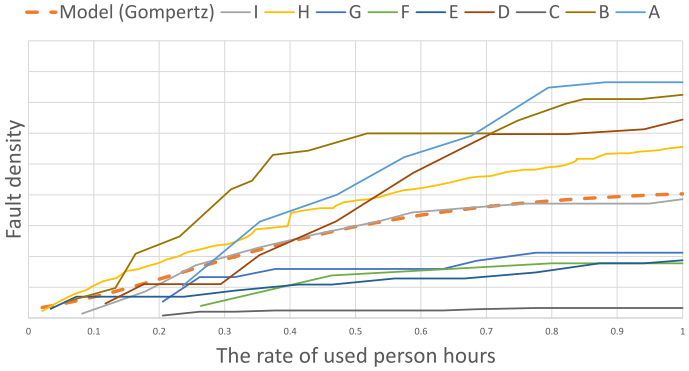

Fig. 7. Results of the fault densities and the rates of used person hours.

5) Rates of tested cases: Figure 8 shows the results of the model using the fault densities and the rate of tested cases, where the x-axis represents the rate of implemented test case and the y-axis represents the fault density. The legend is the same as Figure 7.

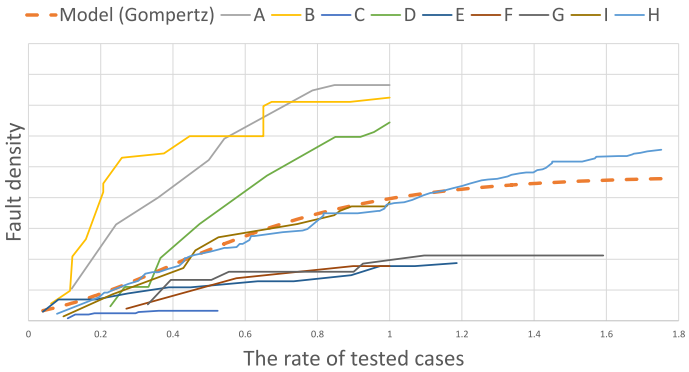

Fig. 8. Results of the fault densities and the rates of used person hours.

B. Discussion

1) RQ1 (Do the results of SRGMs using person hours differ from those using calendar time?): SRGMs based on calendar time and those based on person hours produce different results, implying that SRGMs based on person hours only reflect actual efforts and do not consider non-working time. Table I shows that all the RSSs for all projects differ based on person hours and calendar times, demonstrating that SRGMs based on person hours and calendar time have unique features (RQ1). However, several projects have similar RSSs between calendar time and person hours, indicating that developers worked on the project during holidays and did not stop testing.

2) RQ2 (How do they differ?): SRGMs based on person hours are more precise than SRGMs based on calendar time. For all dataset, the RSSs based on person hours are lower than those based on calendar time. Table I indicates that Project F (B) has the greatest (smallest) decrease of about 97% (13%) when using person hours. For Project F, the RSS of the Logistic model in calendar time is 0.08373, and the RSS of the Logistic model in person hours is 0.002522. The

TABLE I
COMPARISON OF SRGMS BASED ON CALENDAR TIME AND PERSON HOURS

Project	RSS (Calendar Time)		RSS (Person Hours)		Calendar Time / Person Hours	
	Logistic	Gompertz	Logistic	Gompertz	Logistic	Gompertz
A	122.99	115.07	26.99	23.06	0.2194	0.2004
B	130.38	100.66	113.5	85.75	0.8705	0.8519
C	12.82	12.65	6.911	6.317	0.5391	0.4994
D	177.1	NaN	35.26	68.97	0.1991	NaN
E	13.4	12.91	11.57	10.95	0.8634	0.8482
F	0.08373	0.2568	0.002522	0.01944	0.0301	0.0757
G	12.82	12.65	6.911	6.317	0.5391	0.4994
H	3993	2460	2936	1408	0.7353	0.5724
I	34.19	28.75	11.13	4.284	0.3255	0.1490

TABLE II
CORRELATIONS BETWEEN EACH VALUE

	FP	LOC	Fault	Fault/LOC	Fault/FP	Estimated TC	Implemented TC	Calendar time	Person Hours
FP	1	0.9008	0.93883	0.12692	-0.1931	0.7648	0.91526	0.93732	0.98159
LOC	0.9008	1	0.77178	-0.12702	-0.30505	0.63311	0.72619	0.81877	0.83476
Fault	0.93883	0.77178	1	0.42154	0.14561	0.77432	0.93433	0.92996	0.95582
Fault/LOC	0.12692	-0.12702	0.42154	1	0.83274	0.43197	0.39555	0.22485	0.20124
Fault/FP	-0.1931	-0.30505	0.14561	0.83274	1	0.02498	0.01007	-0.05423	-0.1113
Estimated TC	0.7648	0.63311	0.77432	0.43197	0.02498	1	0.93057	0.76607	0.7607
Implemented TC	0.91526	0.72619	0.93433	0.39555	0.01007	0.93057	1	0.90194	0.92417
Calendar time	0.93732	0.81877	0.92996	0.22485	-0.05423	0.76607	0.90194	1	0.9478
Person hours	0.98159	0.83476	0.95582	0.20124	-0.1113	0.7607	0.92417	0.9478	1

TABLE III
COMPARISON OF SRGMS BASED ON PERSON HOURS AND IMPLEMENTED TEST CASES

Project	RSS (Person Hours)		RSS (implemented test cases)		Person Hours / implemented test cases	
	Logistic	Gompertz	Logistic	Gompertz	Logistic	Gompertz
A	26.99	23.06	20.59	15.73	1.170	1.120
B	113.5	85.75	200.6	176.9	1.324	0.427
C	6.911	6.317	6.852	6.200	1.094	0.922
D	35.26	68.97	29.52	25.31	0.511	2.336
E	11.57	10.95	15.33	13.94	1.056	0.714
F	0.002522	0.01944	0.02495	0.06991	0.1297	0.7792
G	6.911	6.317	6.852	6.200	1.094	0.9219
H	2936	1408	4095	2469	2.085	0.3438
I	11.13	4.284	13.27	9.947	2.598	0.3228

value of RSS of person hours divided by calendar time is around 3%. On the other hand, the RSS of the Logistic model in calendar time for Project B is 130.38 and the RSS of the Logistic model in person hours is 113.5. The value of RSS of person hours divided by calendar time is around 87%. These results confirm that there is difference between SRGMs based on calendar time and person hours (RQ2).

3) RQ3 (Do specific metrics evaluate progress?): The number of faults is significantly related to person hours and tested cases in the nine project datasets, suggesting that the person hours required to determine the number of faults can be modeled. The largest correlation coefficient with the number of faults is Person Hours, which is 0.95582 followed by Function Point (0.93883), Implemented test cases (0.93433), and Calendar time (0.92996). Except for Function Point, we monitored the values as a time series. Because the Function Point occurs at the beginning of development, estimating the number of faults is useful using Function Points. Thus, specific metrics can be used to evaluate progress (RQ3).Because the fault density and rates of used person hours are related, SRGMs based on the faults density, rate of used person hours, and rate of used tested cases provide better fitting models than those based on calendar time. Moreover, the nine managers that we interviewed indicated that the leveled lines should assist in confirming progress. Table III shows that the fitness of SRGMs based on person hours and implemented test cases depends on the project. For Project I, the RSS of the Logistic model in person hours is 11.13, and the RSS of the Logistic model in implemented test cases is 13.27. The value of the RSS of person hours divided by implemented test cases is around 260%, which is the largest rate in Table III.

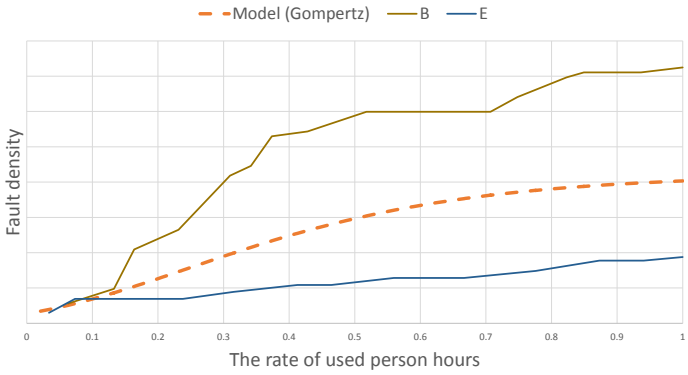

Fig. 9. Fault densities and rates of used person hours for project B and E and the leveled Gompertz model

4) RQ4 (Can progress of projects be directly compared?): Figure 2 shows an example of the fault densities and the rates of person hours for Projects B and E as well as the leveled lines of the Gompertz model through nine projects. Although the two projects cannot be compared in Figure 9 due to differences in the x- and y-axes, the axes in Figure 2 are the same. Additionally, leveled lines can be prepared for the projects, demonstrating that the progress of different projects can be directly compared (RQ4).

Moreover, we have introduced a system for using our method in Sumitomo Electric Industries, Ltd. and made a request to several managers and developers for using our method. After several months, we had interviewed the managers and developers who had used our method for about 3 hours. Our interviews about the models indicate that leveled SRGMs provide useful information about the progress of the projects.

C. Limitations

1) Internal validity threats: In the comparison, we used nine datasets from the same company. Therefore, the data may contain mistakes or other false elements. Moreover, the data contains several domains.

2) External validity threats: We only tested SRGMs based on person hours with nine datasets, which is insufficient to make generalizations about SRGMs based on person hours.

V. Related work

Many different types of software reliability growth models exist. Yamada *et al*. proposed an extend NHPP model, which is related to the test-domain dependence [19]. Test-domain dependent models include 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.

Fujii *et al*. developed a quantitative software reliability assessment method via incremental development processes, which is a type of agile software development based on familiar non-homogeneous Poisson processes [20]. Fujii *et al*. used both the number of faults and software metrics to demonstrate software reliability predictions via a case study.

Several researchers study about the cross project fault prediction and management. Zimmermann *et al*. studied cross-project defect prediction models in points of lines of codes and bugs and other factors [21]. For 12 real-world applications including open source software and enterprise software, they ran cross-project predictions using finished projects datasets. In this paper, we focus on the time series of projects and monitoring the projects in points of person hours and implemented test cases.

Kuo *et al*. proposed a new scheme for constructing software reliability growth models based on NHPP [22]. Their scheme providea a model which considers testing efforts and fault detection rates. They estimate the testing efforts and predict the trends of fault detection rates which can be obtained from historical records of previous releases or other similar software projects. Our method did not need the estimation of testing efforts and used the projects developed by the same company and using the same framework.

VI. Conclusion

Using SRGMs based on person hours, we successfully modeled nine actual datasets. SRGMs based on person hours can more precise model the datasets compared to SRGMs based on calendar time. SRGMs based on person hours are between 13% and 97% more precise than those based on calendar time.

Moreover, we propose leveled SRGMs based on the fault density and the rates of person hours as well as the rates of used test cases. Our interviews of managers about the models indicate that leveled SRGMs provide useful information about the progress of the projects.

References

[1] A. Goel, "Software reliability models: Assumptions, limitations, and applicability," *Software Engineering, IEEE Transactions on*, vol. SE-11, no. 12, pp. 1411–1423, Dec 1985.

[2] 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.

[3] 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.

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

[5] 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.

[6] K. Honda, H. Washizaki, and Y. Fukazawa, "A generalized software reliability model considering uncertainty and dynamics in development," in *Product-Focused Software Process Improvement*, ser. Lecture Notes in Computer Science, J. Heidrich, M. Oivo, A. Jedlitschka, and M. Baldassarre, Eds. Springer Berlin Heidelberg, 2013, vol. 7983, pp. 342–346.

[7] K. Honda, H. Washizaki, and Fukazawa, "Predicting release time based on generalized software reliability model (gsrm)," in *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*. IEEE, 2014, pp. 604–605.

[8] H. Washizaki, K. Honda, and Fukazawa, "Predicting release time for open source software based on the generalized software reliability model," in *Agile Conference (AGILE), 2015*, August 2015.

[9] K. Honda, H. Nakai, H. Washizaki, Y. Fukazawa, K. Asoh, K. Takahashi, K. Ogawa, M. Mori, T. Hino, Y. HAYAKAWA *et al.*, "Predicting time range of development based on generalized software reliability model," in *21st Asia-Pacific Software Engineering Conference (APSEC 2014)*, 2014.

[10] T. Dohi and N. Nakagawa, "Stochastic reliability and maintenance modelling," 2013.

[11] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE transactions on Reliability*, vol. 3, pp. 206–211, 1979.

[12] M. Anjum, M. A. Haque, and N. Ahmad, "Analysis and ranking of software reliability models based on weighted criteria value," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 5, no. 2, p. 1, 2013.

[13] 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.

[14] "The r project for statistical computing," `http://www.r-project.org/`.

[15] 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.

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

[17] 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.

[18] 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.

[19] S. Yamada, H. Ohtera, and M. Ohba, "Testing-domain dependent software reliability models," *Computers and Mathematics with Applications*, vol. 24, no. 12, pp. 79 – 86, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0898122192902316

[20] T. Fujii, T. Dohi, and T. Fujiwara, "Towards quantitative software reliability assessment in incremental development processes," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 41–50. [Online]. Available: http://doi.acm.org/10.1145/1985793.1985800

[21] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ser. ESEC/FSE '09. New York, NY, USA: ACM, 2009, pp. 91–100. [Online]. Available: http://doi.acm.org/10.1145/1595696.1595713

[22] S.-Y. Kuo, C.-Y. Huang, and M. R. Lyu, "Framework for modeling software reliability, using various testing-efforts and fault-detection rates," *IEEE Transactions on Reliability*, vol. 50, no. 3, pp. 310–320, Sep 2001.