Journal of Software Engineering and Applications, 2014, 7, 396-405
Published Online May 2014 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2014.75036

A New Analysis Concept in Applying Software Reliability Growth Models and Tool Implementation: The SafeMan

Takaji Fujiwara[1], Mitsuhiro Kimura[2]

[1]SRATECH Laboratory Inc., Hyogo, Japan
[2]Department of Industrial and Systems Engineering, Faculty of Science and Engineering, Hosei University, Tokyo, Japan
Email: fujiwara.takaji@nifty.com, kim@hosei.ac.jp

Received 20 February 2014; revised 18 March 2014; accepted 25 March 2014

Abstract

In recent years, many software development organizations have been assessing and analyzing their software product's reliability/quality and judging whether the software product is releasable by using Software Reliability Growth Models (SRGMs) at the final stage of software development. The usage of SRGMs originates in the advantage that various reliability analysis results based on the SRGMs can be acquired easily. However, it is very difficult for general software project managers to grasp the achievement level of reliability/quality based on its analysis results because some sort of professional knowledge is required in order to understand the information on the attainment progress of software product's reliability/quality. Moreover, it is also difficult for software project managers and inspectors who do not deeply comprehend the details of their project to evaluate the degree of software reliability and quality, if they assess it without grasping the live development situation and only see the documents submitted from their staff. In this paper, we propose a new analysis concept for assessing the software product's reliability/quality, and illustrate the output results obtained by a tool, the SafeMan.

Keywords

Software Reliability Growth Model, Time Series Analysis, Software Reliability Tool, SafeMan

1. Introduction

In the highly informational society, software products have been playing an integral part and important role of the computer systems, information communication technology (ICT) systems, embedded systems, and so on. Accordingly, since the occurrences of software failures due to the software product can cause severe consequences, the reliability of the software product is a primary concern for both software developers and its users.

For that reason, the testing process which is located in the final stage of a waterfall-type software development is an important activity for detecting and removing the latent faults and improving the reliability/quality of the software product. However, because it is impossible to detect and remove all the faults latent in the software product within a limited testing period, we need a quantitative method for assessing and analyzing the reliability/quality during the testing phase of the development. As a method for solving this problem, many Software Reliability Growth Models (abbreviated as SRGMs) [1] [2] which involve the testing- and operational-environment factors have been proposed so far. In particular, it is increasing the case where the project managers have used the reliability assessment results derived by SRGMs in order to determine when to stop testing and release the software product as an important management issue during the testing. Consequently, by visualizing these reliability analysis results, many software tools (SORPS [3], SRET [4], SREPT [5], SafeMan [6], SRATS [7], etc.) which enable project managers to easily understand the achievement level of the software reliability have been developed until now. It is known that these tools have been applied in the actual software development projects. Furthermore, Fujiwara *et al*. [8] have proposed the SafeMan tool which not only shows the analysis results of the specified reliability assessment measures, but also displays the easy messages which an expert analyzed based on the visualized results.

However, these tools provide only the software reliability/quality analysis results based on the inputted analytical data as taking a snapshot of the development. Upon using the conventional tools or analysis methods, it is difficult to judge whether software reliability/quality is growing by the current testing-activities. It is also a hard issue for a manager who does not grasp the details of a project to judge the project tendency.

In other words, using the software development tools which take only snapshots of the development does not yield sufficient information on the software management. The users of the tools need to read more carefully the reliability assessment results obtained from the tools by a view point of the tendency analysis like the time series analysis. Grasping the tendency of the software reliability and quality along with the progress of the development will really help the managers.

In this paper, we illustrate four kinds of graphs in order to grasp the tendency of the software development process by using analysis results of past-records information as a method of solving these difficulties on software management mentioned above. Actually we have implemented the structure which accumulates the analysis results by drawing these trend-analysis graphs into the SafeMan tool. The SafeMan tool can provide the information on the attained software reliability in the software testing phase taken along our concept discussed in this paper.

Hence we explain the fundamental functions on the SafeMan tool in Section 2. That is, we describe the SRGMs and some assessable measures, and the outline of the functions supporting the project- and testing-managers, which have been implemented in the SafeMan tool. In Section 3, we discuss the four graphs which were obtained as the reliability analysis results by using the datasets in terms of software reliability, from the point of view of how we can understand the development tendency by the graphs.

Finally, we also illustrate several examples of the processing sequence of the SafeMan tool, its data input screen, and the output results (including tendency graphs) which have implemented the functions discussed in Section 3.

2. The SafeMan Tool

In this section, we explain the fundamental functions of the SafeMan tool which has been developed for the purpose of easy-to-use for the general project- and testing-managers.

2.1. Implemented SRGMs

First, we briefly show a mathematical background of SRGMs, which were implemented into the tool in order to provide the reliability/quality assessment results with high accuracy for the software quality managers, based on the fault data obtained during the testing phase. The implemented tool, the SafeMan is expected to play a role of the specialized software development staff.

Let $N(t)$ be the random variable denoting the cumulative number of faults detected up to testing time ($t \geq 0$).

$H(t)$ is the expectation of $N(t)$. That is $H(t)$ is called the mean value function representing the expected cumulative number of faults detected up to testing time *t*. Then, an SRGM based on a nonhomogeneous Poisson process (abbreviated as NHPP) [1] [2] is formulated by specifying the mean value function as:

$$\Pr\left[N(t)=n\right]=\frac{H(t)^{n}}{n!}\exp\left[-H(t)\right]\left(n=0,1,2,\cdots\right)$$
$$H(t)=\int_{0}^{t}h(x)\,\mathrm{d}x \tag{1}$$

where $\Pr[\mathrm{A}]$ means the probability of the event $\{\mathrm{A}\}$ and $h(t)$ is called the intensity function of an NHPP, which represents the instantaneous fault-detection rate at testing time *t*.

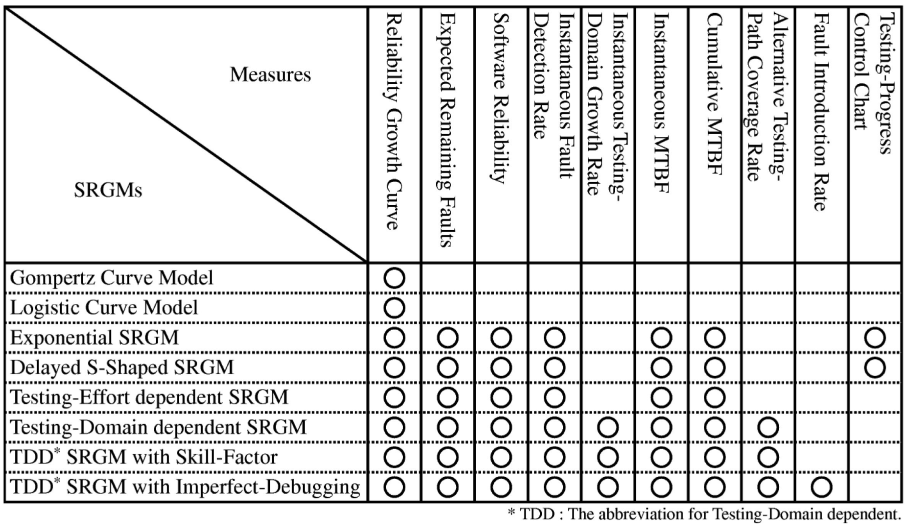
In the literature, a software reliability growth means the relationship between the testing time and the cumulative number of faults detected by the testing. Therefore the reliability growth curve represents the time-dependent behavior of the cumulative number of detected faults with the progress of the testing time. This property is a baseline of the time series analysis of software reliability evaluation by the SafeMan. The SafeMan tool provides six NHPP models: two basic NHPP models [1] [2] and four extended NHPP models [2] [9]. Also the SafeMan is implemented the Gompertz and Logistic curve models which are called deterministic regression models. These models are not included in the NHPP models. Historically, these models have been widely applied to practical project data for the software quality/reliability assessment by many main-framers and software industries in Japan [2]. Furthermore, these deterministic models can estimate the total number of faults latent in the software system as the convergence value of the Gompertz and Logistic curve models by using the nonlinear regression analysis. Although these two models do not consider any stochastic properties in the relationship between the testing time *t* and the cumulative number of detected faults, they belong to SRGMs.

The list of SRGMs and the reliability assessment measures implemented into the SafeMan tool is shown in **Figure 1**. The round mark in **Figure 1** represents the measure which can be assessed by each SRGM.

2.2. SafeMan's Functions [8]

Next, we describe main two functions implemented in the SafeMan which support the general project- and testing-managers, *i.e.*, the automatic selection of the optimum SRGM, and the automatic derivation of testing-management policies.

The SafeMan provides SRGMs which describe plural competing-model phenomena, in order to realize high accurate prediction. For this reason, we are often confronted with the problem that we cannot determine the suitable SRGM for reliability/quality assessment of the software product during the testing phase, based on the project- and testing-manager's skill. That is, although the special knowledge is needed when determining the



SRGMs \ Measures	Reliability Growth Curve	Expected Remaining Faults	Software Reliability	Instantaneous Fault Detection Rate	Instantaneous Testing-Domain Growth Rate	Instantaneous MTBF	Cumulative MTBF	Alternative Testing-Path Coverage Rate	Fault Introduction Rate	Testing-Progress Control Chart
Gompertz Curve Model	○									
Logistic Curve Model	○									
Exponential SRGM	○	○	○	○		○	○			○
Delayed S-Shaped SRGM	○	○	○	○		○	○			○
Testing-Effort dependent SRGM	○	○	○	○		○	○			
Testing-Domain dependent SRGM	○	○	○	○	○	○	○	○		
TDD* SRGM with Skill-Factor	○	○	○	○	○	○	○	○		
TDD* SRGM with Imperfect-Debugging	○	○	○	○	○	○	○	○	○	

* TDD : The abbreviation for Testing-Domain dependent.

Figure 1. List of the SRGMs and their assessment measures implemented into the SafeMan tool.

best-fit SRGM to the fault data obtained, few general project- and testing-managers usually have such knowledge.

Thus, the general project- and testing-managers can obtain the selection skill of the best SRGM at the expert level by using the SafeMan. For this purpose, SafeMan calculates the average sum of square-errors and the Akaike Information Criterion [10] which are both widely-known as the goodness-of-fit criteria for determining the optimum SRGM.

On the other functions, the NHPP models and deterministic models can help project- and testing-managers by providing useful quantitative measures for the reliability/quality assessment of the software product. Although we can derive easily the testing-management policy from the information on these assessment measures by reading technical books, it is very rare case that quite the same analysis results are obtained in all assessment measures. That is, in the actual situation, understanding the reliability/quality evaluation results is complicated even if the managers use the software reliability tools. To overcome this issue, the SafeMan supports the function which displays analysis results with easy messages based on an expert person's know-how in terms of the appropriate testing-management policy selection.

3. Grasping the Development Process Tendency by Time Series Analysis

In this section, we describe methodologies for grasping and judging the time-dependent behavior of software product's reliability/quality based on the past analyzed records during the testing phase by a management group who does not grasp the details of a project, or the inspection people (we use the term *inspectors* in the following) who have to judge the assigned software product based on only the given documents from the supplier (e.g. software venders, etc.).

This is the method that we can grasp the tendency when the analyst records information shown in the following based on SRGMs as the past quantitative analysis results of reliability/quality performed during the testing.

- Estimated total number of faults
- Estimated software reliability
- Estimated number of detectable faults
- Estimated optimal release date based on two criteria

One who cannot grasp the project in detail is able to judge the testing-activities (as good or bad) performed up to now by drawing graphs which recorded the above-mentioned analysis results for every analysis.

In the following, we explain four kinds of graphs in order to grasp the tendency of the achieved quality by using the past analysis results records.

3.1. Time-Dependent Behavior of the Total Number of Faults

Figure 2 is a graph which shows the time-dependent behavior of the total number of faults [1] [2] obtained at each analysis time point. The figure has the analysis-execution date on the horizontal axis and the total number of detected and removed faults on the vertical one based on the analysis results obtained by the tool.

We can judge whether the reliability of the developed software reaches at the sufficient level and can end the testing by seeing the fluctuating behavior of the estimated total number of detected faults, even if the project members and the management group who do not grasp the details of the project. Recall that this is the tendency analysis mentioned in the previous section.

Also, by analyzing this graph submitted from the supplier, the inspector can grasp the software product's reliability/quality attainment level, and can judge whether receiving the software is possible or not. That is, at the State 1 in **Figure 2**, we have to determine that the testing is still required, because the fluctuation of the estimated total number of faults (width of increase and decrease) is large for every analysis. Therefore, the inspector must judge that the software product should not be received from the supplier. On the other hand, at the State 2 in **Figure 2**, we can see that it is about the time when we need to judge whether we can stop the testing and also the software product can be received from the supplier, because the behavior of the estimated total number of faults based on the analysis results is stable on a certain value, without being influenced by the testing-activity.

Consequently, if the analysis results of the past records converged to a certain value on the graph, we can consider that the testing-activity under execution is going to the good direction (*i.e.*, the reliability/quality is growing), and can suggest that the adequate testing has been performed.

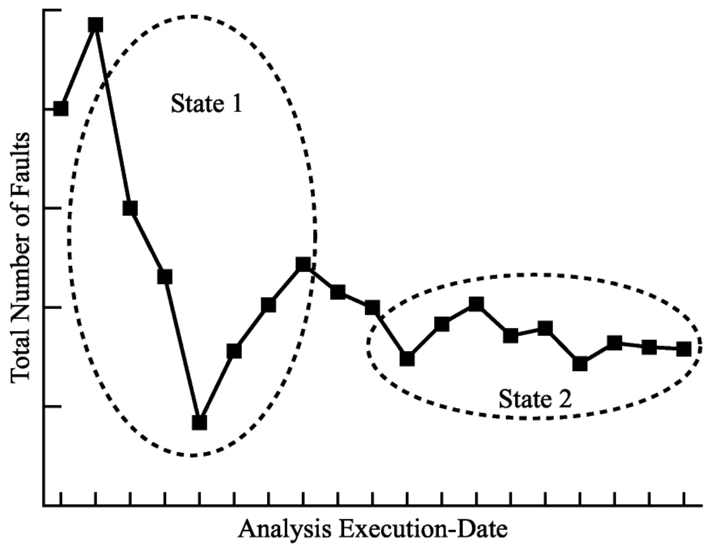


Figure 2. Time-dependent behavior of the estimated total number of faults.

3.2. Time-Dependent Behavior of the Software Reliability

Figure 3 shows the time-dependent behavior of the estimated software reliability which is the probability that the software is operating according to the customer requirement without any failure occurrence by the testing for the specified day [1]. **Figure 3** has the analysis-execution date on the horizontal axis and the software reliability based on the analysis results on the vertical axis. Even if the project members and the management group do not comprehend the details of the project, they can know whether the software product is steadily operating by seeing the graph of the software reliability. Also, the inspector can see whether the software product is receivable by analyzing this graph submitted from the supplier.

Hence by using such graphical illustrations generated by the software reliability tool, we can grasp that the testing activity under execution is going to the good direction (*i.e.*, the reliability/quality is growing), and we can evaluate that the adequate testing has been performed.

3.3. Time-Dependent Behavior of the Number of Detectable Faults

Figure 4 shows the time-dependent behavior of the degree of increase and decrease of the number of *detectable* faults a day by the software testing. In **Figure 4**, the horizontal axis represents the analysis-execution date and the vertical one indicates the number of detectable faults based on the analysis results. Here, we explain the number of detectable faults. In the testing phase, we execute many various test-cases in order to verify the implemented functions based on the customer requirements. Then, there exists a set of the testing-paths in the software to be influenced by executing test-cases, and the performance of these testing-paths in the software product is represented as the testing-path coverage. Thus, the latent software faults covered by these testing-paths are called the detectable faults. That is, we consider that one fault may be latent in each newly covered testing-path by the testing a day. Then, the number of detectable faults increases, and it can be given by the instantaneous testing-domain growth rate [2] which is one of the reliability assessment measures.

On the other hand, when the testing activity detects several software faults a day, the results can be shown by the instantaneous fault-detection rate [1] [2] included in the reliability assessment measures. Then, the number of detectable faults decreases because some of detectable faults were actually detected and removed. Thus, we can judge whether the software reliability/quality has achieved the level at which the project members and the management group who do not grasp the details of project can end the testing by considering the difference between the instantaneous fault-detection rate and the instantaneous testing-domain growth rate.

Also, by analyzing this graph (**Figure 4**) which was submitted from the supplier, the inspector can see the software product's reliability/quality attainment level, and can judge whether receiving of the software product is possible. In **Figure 4**, we have to make our decision that some additional testing is still required when we are at the State 1, because the variation of the number of detectable faults (width of increase and decrease) is large

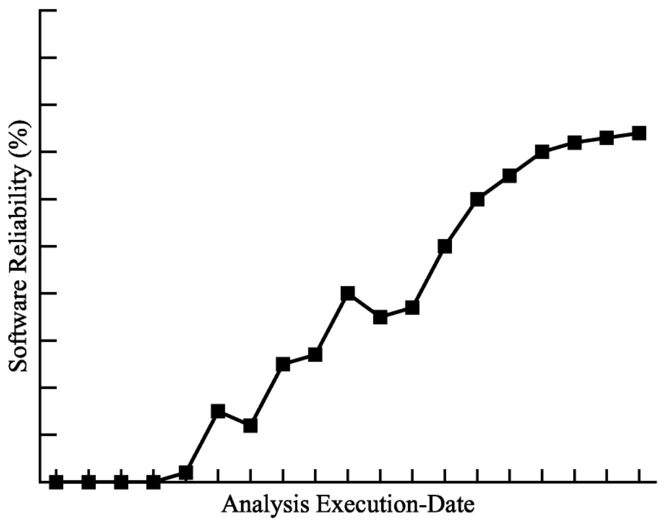


Figure 3. Time-dependent behavior of the estimated software reliability.

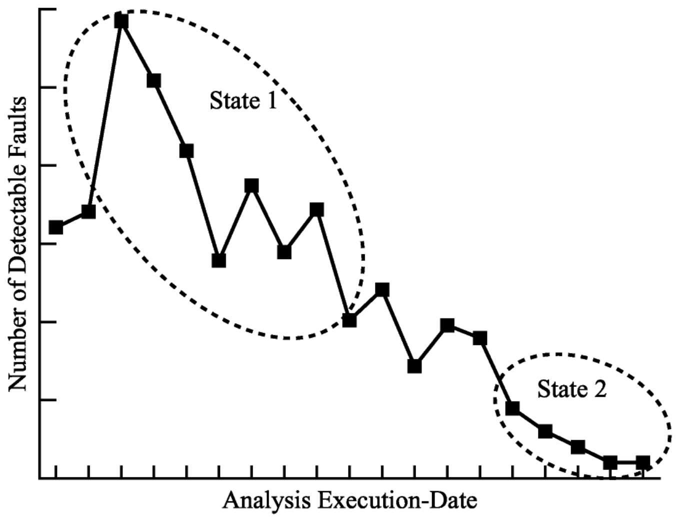


Figure 4. Time-dependent behavior of number of detectable faults.

for every analysis. Therefore, the inspector should judge that the software product cannot be received from the supplier. On the other hand, if we are at the State 2 in Figure 4, we find that it is the time when we can stop the testing, and also we can receive the software product from the supplier, because the estimated number of detectable faults based on the analysis results is close to zero and it is in the stable behavior.

Accordingly, if the analysis results of the past records can show the graphs converging to zero (the objective value) we can consider that the testing-activity under execution is progressing to the good direction (*i.e.*, the reliability/quality is growing), and such results suggest that the adequate testing process has been performed.

3.4. Time-Dependent Behavior of the Releasable Date

Also Figure 5 illustrates the time-dependent behavior of two kinds of predicted releasable dates, *i.e.*, the achievable date to the aim of the remaining fault rate (black solid line) and the optimal date minimizing the total software cost [1] [2] (gray solid line). Figure 5 sets the analysis-execution date on the horizontal axis and the releasable date based on the analysis results on the vertical axis. We explain the remaining fault rate. In the software testing processes, everyone knows well the fact that it is impossible to remove all faults latent in the software. Therefore, considering the characteristics of the customer requirements and the software product, we

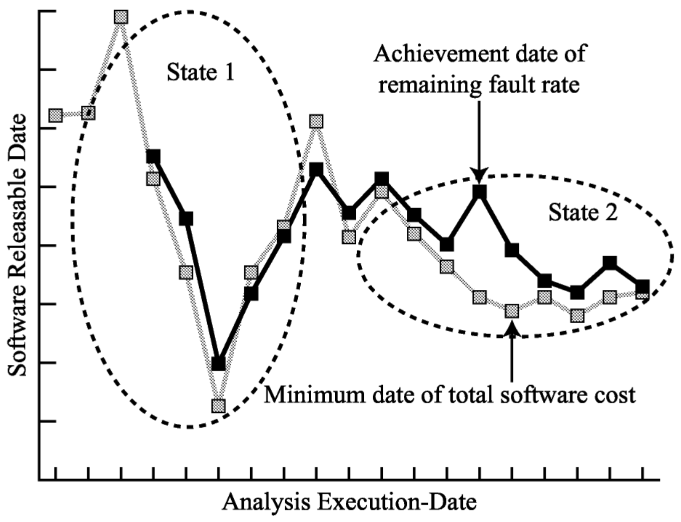


Figure 5. Time-dependent behavior of the estimated releasable date.

can set up the target value for the number of faults which should be detected by defining the allowable value of remaining fault detection rate for the software. That is, the achievement date of the objective remaining fault rate means the date that the total number of faults estimated by the analysis will reach below the specified allowable value of the remaining fault rate. Therefore the date of minimizing the total software cost is the time that the predicted total life-cycle cost of the software based on the debugging and maintenance cost becomes the minimal value [1] [2]. Hence we can estimate the date when to stop the testing and release the software product to the customer or which we can receive the software product from the supplier, based on both assessment measures.

At the State 1 in **Figure 5**, we have to judge that the testing is still more required, because the releasable date has large fluctuation (on the amplitude of the graph) according to the variation of the total number of faults in the software product for every analysis. Therefore, the inspector should determine that the software product cannot be received from the supplier. On the other hand, at the State 2 in **Figure 5**, we find that it is the time when we should judge whether we can stop the testing, or the software product can be received from the supplier, because the releasable date goes to the stable behavior on a certain date without any large fluctuation.

Thus, when such sequential analysis results of the past records become stable, we can think that the testing-activity under execution is proceeding to the good direction and the results can suggest us that the adequate testing has been performed.

Note that when the convergence dates of both assessment measures differ, we need to adjust the values of cost parameters included in the software cost model implemented in the software tool, so as to be the same convergence behavior.

4. Drawing the Tendency Graphs in SafeMan

In this section, we discuss the requirement specification definitions for the implementation of the drawing function of four kinds of the tendency graphs discussed in Section 3 by slightly modifying the SafeMan tool. We illustrate the data input screen, analysis results of the past-records for each analysis day, and four kinds of tendency graphs based on these data in the implementation of the requirement specification definitions for the SafeMan. The requirement specification definitions of the graph drawing functions for the SafeMan are listed as follows:

1) To reuse the basic functions of the current version of the SafeMan

2) To specify the date which indicates the analysis result in data input screen

3) To be able to specify whether analysis results records or not

4) To output the reliability analysis results of the analysis day to the Excel sheet

5) To plot four kinds of graphs using the Excel's graph drawing functions

The outline of the data processing sequence in the SafeMan based on these requirement specification definitions above is shown in **Figure 6**. **Figure 6** depicts the SafeMan data input screen implemented the above items

(2) and (3), and four kinds of graphs in order to analyze the tendency by using the output results of the reliability estimation. Also Excel's graph drawing function from the SafeMan which implements the requirement specification definition of items (4) and (5) is illustrated in **Figure 7**.

In **Figure 7**, among the controls of the SafeMan window, a check button labeled *Outputpast-records* means that the SafeMan validates the data with round mark "○" in the fault data list. When round mark is put on the

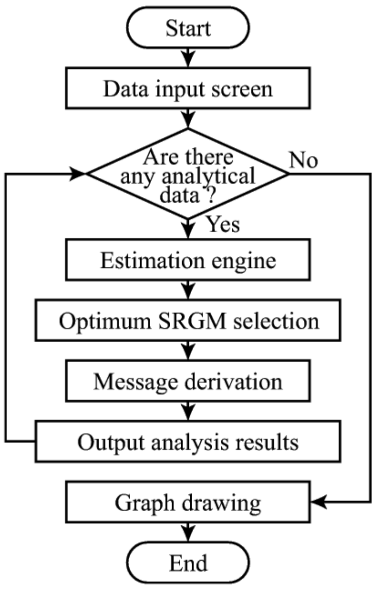
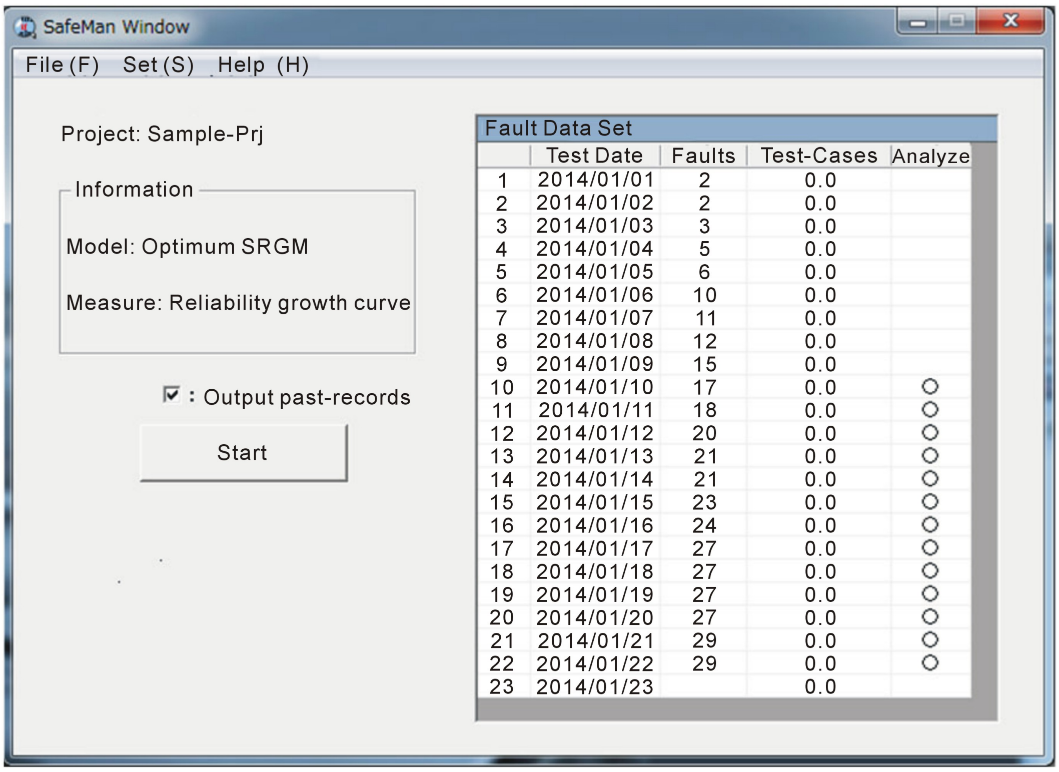


Figure 6. Procedure for obtaining tendency graphs based on the analysis of the past records.



Project: Sample-Prj

Information

Model: Optimum SRGM

Measure: Reliability growth curve

☑ : Output past-records

Fault Data Set

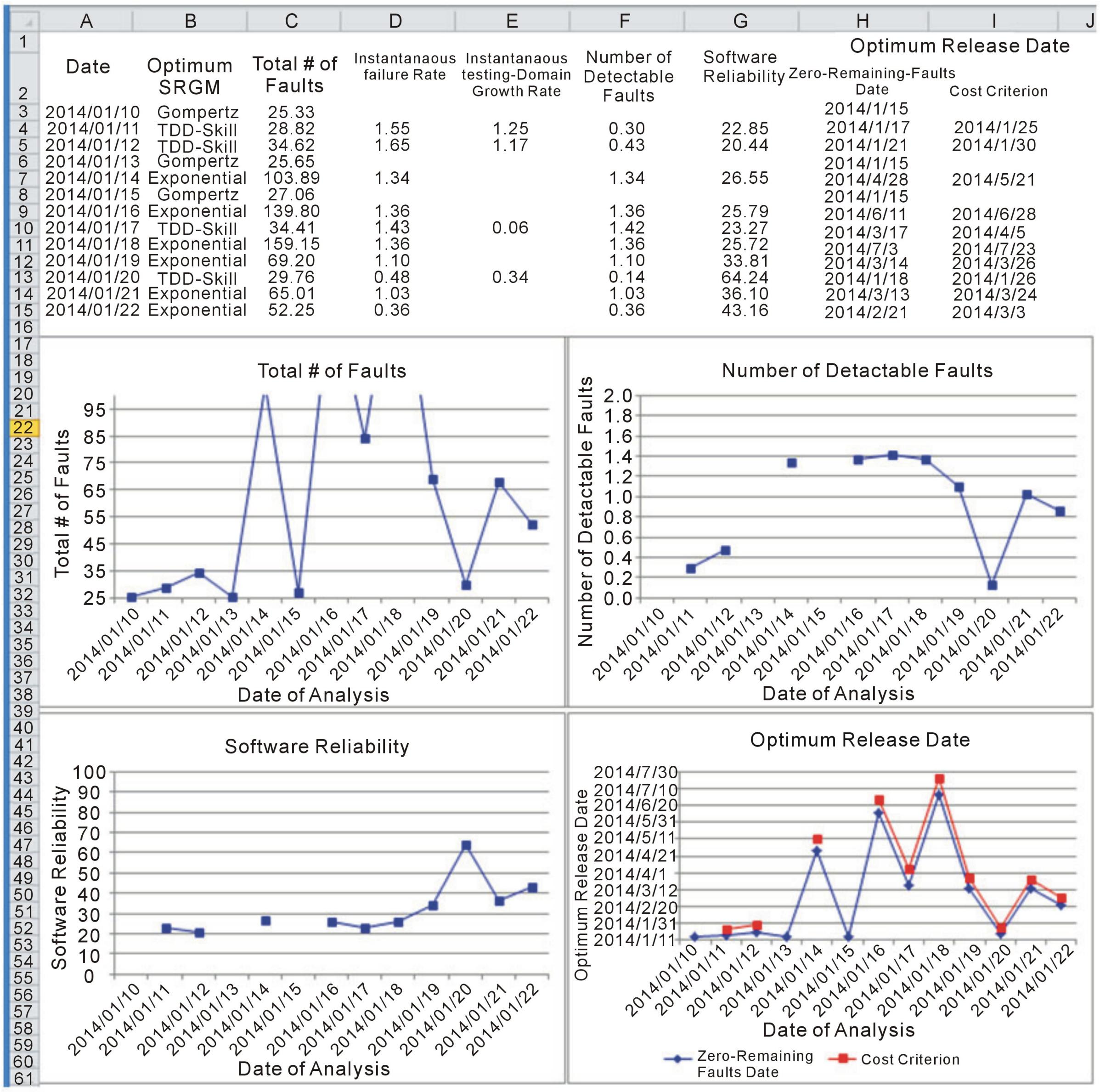
	Test Date	Faults	Test-Cases	Analyze
1	2014/01/01	2	0.0	
2	2014/01/02	2	0.0	
3	2014/01/03	3	0.0	
4	2014/01/04	5	0.0	
5	2014/01/05	6	0.0	
6	2014/01/06	10	0.0	
7	2014/01/07	11	0.0	
8	2014/01/08	12	0.0	
9	2014/01/09	15	0.0	
10	2014/01/10	17	0.0	○
11	2014/01/11	18	0.0	○
12	2014/01/12	20	0.0	○
13	2014/01/13	21	0.0	○
14	2014/01/14	21	0.0	○
15	2014/01/15	23	0.0	○
16	2014/01/16	24	0.0	○
17	2014/01/17	27	0.0	○
18	2014/01/18	27	0.0	○
19	2014/01/19	27	0.0	○
20	2014/01/20	27	0.0	○
21	2014/01/21	29	0.0	○
22	2014/01/22	29	0.0	○
23	2014/01/23		0.0	

Figure 7. Data input screen of the SafeMan tool.

analysis column in the fault data list like the figure, SafeMan performs the reliability analysis from the 1st day of the testing to the day with the round mark. Then the SafeMan outputs the information obtained to the Excel as the analysis results. This analysis is repeated and carried out by the number of times attached the round marks. This analysis continues to the last inputted data, the graphs based on the analysis results are drawn and the data processing on the SafeMan is completed (see **Figure 8**). Consequently, the SafeMan has the functions which help the software development managers to investigate the tendency analysis by giving the time-dependent reliability and quality analysis results of the four aspects.

5. Concluding Remarks

In this paper, we have discussed the visualization technology and the decision-making method of reliability/quality of the software product as a new analysis concept along with several illustrations. In particular, we have shown that the software management group who does not (or, cannot) pay attention to the details of the assigned software projects or the inspector who has to judge about receiving the software product based on the



	A	B	C	D	E	F	G	H	I
1								Optimum Release Date	
2	Date	Optimum SRGM	Total # of Faults	Instantanaous failure Rate	Instantanaous testing-Domain Growth Rate	Number of Detectable Faults	Software Reliability	Zero-Remaining-Faults Date	Cost Criterion
3	2014/01/10	Gompertz	25.33					2014/1/15	
4	2014/01/11	TDD-Skill	28.82	1.55	1.25	0.30	22.85	2014/1/17	2014/1/25
5	2014/01/12	TDD-Skill	34.62	1.65	1.17	0.43	20.44	2014/1/21	2014/1/30
6	2014/01/13	Gompertz	25.65					2014/1/15	
7	2014/01/14	Exponential	103.89	1.34		1.34	26.55	2014/4/28	2014/5/21
8	2014/01/15	Gompertz	27.06					2014/1/15	
9	2014/01/16	Exponential	139.80	1.36		1.36	25.79	2014/6/11	2014/6/28
10	2014/01/17	TDD-Skill	34.41	1.43	0.06	1.42	23.27	2014/3/17	2014/4/5
11	2014/01/18	Exponential	159.15	1.36		1.36	25.72	2014/7/3	2014/7/23
12	2014/01/19	Exponential	69.20	1.10		1.10	33.81	2014/3/14	2014/3/26
13	2014/01/20	TDD-Skill	29.76	0.48	0.34	0.14	64.24	2014/1/18	2014/1/26
14	2014/01/21	Exponential	65.01	1.03		1.03	36.10	2014/3/13	2014/3/24
15	2014/01/22	Exponential	52.25	0.36		0.36	43.16	2014/2/21	2014/3/3

Figure 8. Results of the accumulated past-records and their tendency graphs.

only submitted documents from the supplier, can observe the progress tendency (reaching to the aim or failing the achievement) of the software development project by applying the new analysis methodology discussed in Section 3.

Consequently, we have proposed four kinds of tendency evaluation methodologies represented in the software product's reliability/quality in order to make the correct decision with ease by the SafeMan. These graphs can clearly indicate whether the current testing activities contribute to raising the software quality to the sufficient level, by showing the time-dependent behavior of the past-records analysis results.

Furthermore, in order to simply draw four kinds of tendency graphs based on the past-records analysis results of the software reliability/quality assessment which were mentioned above, we have additionally implemented the existing SafeMan tool, and we have illustrated the output results.

For the future studies, we are going to verify the validity of applying this concept and the SafeMan tool to the actual projects more and more. In the actual situations, there are many cases that we have to determine the optimal release date of the software product without grasping the project-activities. Moreover, we need to consider-the operability and the shortening of analysis time of the SafeMan tool, in order to apply the tool to many projects more widely.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 23510189.

References

[1] Pham, H. (2000) Software Reliability. Springer-Verlag, Singapore.

[2] Kimura, M. and Fujiwara, T. (2011) Software Reliability. JUSE Press, Tokyo.

[3] Ohba, M. (1984) Software Reliability Analysis Models. *IBM Journal of Research and Development*, **28**, 428-443. http://dx.doi.org/10.1147/rd.284.0428

[4] Yamada, S., Isozaki, R. and Osaki, S. (1989) A Software Reliability Evaluation Tool: SRET. *Transaction on IEICE*, **J72-D-1**, 24-32.

[5] Ramani, S., Gokhale, S. and Trivedi, K.S. (1998) SREPT: Software Reliability Estimation and Prediction Tool. *Performance Evaluation*, **39**, 37-60.

[6] Fujiwara, T., Yamada, S. and Shiotani, K. (2000) A Software Testing-Management Tool for Reliability of Software Development and Its Application. *Proceedings of the* 6*th International Conference on Reliability and Quality in Design*, Orland, 175-179.

[7] Okamura, H., Ando, M. and Dohi, T. (2005) Development of a Software Reliability Assessment Tool on Spreadsheet Software. *Transaction on IEICE*, **J88-D-1**, 205-214.

[8] Fujiwara, T. and Yamada, S. (2004) Testing-Management Policies by Using the Software Reliability Assessment Tool. *Proceedings of the* 2*nd International Conference on Project Management*, Chiba, 403-408.

[9] Yamada, S. and Fujiwara, T. (2001) Testing-domain Dependent Software Reliability Growth Models and Their Comparisons of Goodness-of-Fit. *International Journal of Reliability, Quality and Safety Engineering*, **8**, 205-218. http://dx.doi.org/10.1142/S0218539301000475

[10] Akaike, H. (1976) What Is the Information Criterion AIC? *Suri Kagaku*, **14**, 5-11.