Empirical Evaluation of Review Process Improvement Activities with respect to Post-Release Failure

Osamu Mizuno and Tohru Kikuno
Graduate School of Engineering Science, Osaka University
o-mizuno@ics.es.osaka-u.ac.jp

Abstract
In this paper, we analyze empirically software quality changes caused by long-term improvement activities for software review process in a certain company.

The objective of our position paper is to present following three points: (1) the review process improvement activity introduced in a company, (2) confirmation that review improvement is fixed in the company (3) evaluation of the effects with respect to the software quality changes caused by the review improvement.

Generally, the effectiveness of review process is evaluated from the viewpoint of high productivity of the software process. But truely speaking, customers really concern about the quality of final software product rather than productivity. Thus we analyze the quality of the final product(post-release failure), and show that the effectiveness of review process improvement acticities based on the analysis result.

1 Process Improvement

1.1 Overview

In the company to be investigated in this paper, the software engineering process group(SEPG) was established in 1992. Since then the SEPG has conducted two main process improvement activities as follows: establishment and introduction of the standards for managing the software process, and improvement of the review process using the software metrics toward high quality.

At first, the SEPG tried to establish several standards for managing software project, and to put it into practice from 1993. The main purposes of the standards include guiding developers in each project to create the well-formed plan and managers to conduct the project successfully according to a well-formed plan. It is believed (without any proof) in the company that the well-formed plan will derive the accurate estimation for the project cost. Furthermore, it is also believed that the accurate cost estimation will lead the project to the high quality of the product and high productivity of development team. We have already applied the statistical analysis to this improvement activity and shown the correctness of these facts by the test of statistical hypothesis(t-test) with a level of significance 5% in [5].

Next, the SEPG has extensively engaged in the improvement of review process. Generally, the cost of removing faults in the later development phases, such as debug or test phase becomes higher than that in the earlier phase such as design review or coding review phase. So it is strongly recommended to remove faults in the earlier phase. Concerning this fact, the number of faults detected in the review activity is believed to be greater by increasing the amount of review activity. Based on the similar experience and knowledge as mentioned above, the SEPG started the improvement of review process in 1995. The key point of the improvement is implementing effective review process by increasing the amount of efforts for review(especially, code review) and by introducing good guidelines. We will discuss only the second review process improvement in this paper.

1.2 Review process improvement

As mentioned before, reviews include the document review in the design activity and the code review in the coding activity. Generally, it is difficult to derive concrete guidelines or numerical target values for the document review[2, 4]. On the other hand, it is relatively easy to derive them for the code review. The situation is also true for the SEPG's activities, as shown in the past analysis result of the review's effect[7].

Based on the analysis result of the past project data, the SEPG has derived the following guidelines G_1–$G6$ for the review activities:

- G_1 At least the 15% of the total efforts for design and coding activities should be assigned to reviews(the document review and the code review).
- G_2 Reviewers must report the progress using the standard review form at regular intervals.
- G_3 In the design review, the documents should be distributed to all the experts in the company, and then review results should be returned to developers via manager(This design review is called peer review[6]).
- G_4 The coding review should be performed by two or three people, including one person who wrote the code.
- G_5 The review coverage rate for the code review should be about 200 lines of code per hour.
- $G6$ All new codes and changed codes should be reviewed. (Concerning reuse of old codes, reviews are not necessary required.)

Among the guidelines above, G_1 and G_2 are general requirements for reviews(including both the document review and the code review), G_3 is specific for the design review, and G_4, G_5 and $G6$ are only for the code review. As for the G_1, it is difficult to determine the criteria for controlling the review activity.

For example, the number of residual faults estimated by the capture-recapture model is used in [1] as a criteria. However, taking our previous work[7] into consideration, we adopt the numerical value of the review effort as a criteria. The criteria 15% of total effort is also shown in [3].

Before the review process improvement began, the same activities as the guideline G_2, G_3 and G_4 were required to be performed in the review. However, since the guidelines were not yet established, the review did not work effectively in the practical developments. Actually the average review effort was less than 10% of the total design efforts. This value 10% is not sufficient according to the experience in the company.

Then, the SEPG started the improvement of review process in 1995 according to these guidelines. Intuitively speaking, recently these guidelines have been followed in the company and no serious failure reports reach to the SEPG. However any formal or statistical discussions on the efforts by the review process improvement are not yet done. So, we try to analyze empirically its effectiveness using actual 23 project data.

2 Fixing Process Improvement

2.1 Software metrics and assertion

Figure 1 shows a simplified process model and a part of fundamental data set collected at each phase of development.

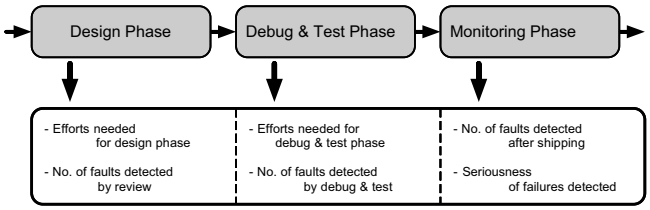

Figure 1: Fundamental data set

As for the efforts spent in the development process, the SEPG collects data of several fundamental metrics. The metrics E_{design} and E_{review} represent the total efforts spent for all activities in design phase and all review activities in design phase, respectively. From the debug & test phase, the SEPG collects data of fundamental metric E_{test}. The metric E_{test} represents the total efforts spent for all activities in the debug & test phase.

In order to evaluate the amount of efforts, we define a metric γ as follows:

$$\gamma_{review/design} = \frac{E_{review}}{E_{design} + E_{review}} \times 100$$

The metric $\gamma_{review/design}$ stands for the ratio of all review effort to design and coding efforts.

We confirm that the review improvement activities are accepted in the development teams. In other words, we investigate the direct changes in the software development process. In more detail, we analyze the changes in the amounts of the effort for review activities. As mentioned in subsection 1.2, the guideline G_1 requires at least 15% of the total efforts on design and coding activities should be spent on review. Thus, we can expect that the effort of review activity is increased in the projects guided by the SEPG as the direct effect by the guideline G_1. Based on these considerations, we try to prove the following assertion A_1 by statistical analysis:

A_1 The ratio of effort for review on the total efforts for design and coding activities increases in the projects guided by the SEPG.

2.2 Ratio of review effort γ

First, concerning the assertion A_1, we try to investigate how the review improvement activities by the SEPG are accepted by the organizations. Here in order to evaluate the process improvement activities, we focus on the project groups rather than individual projects. There exist three project groups: Vending System, Checking System and Retail System.

As mentioned in Section 1, the guideline G_1 recommends that $\gamma_{review/design}$(the ratio of review effort to effort in design phase) should be greater than 15%. According to this guideline, we define a project with $\gamma_{review/design} \geq 15\%$ to be a faithful project. Table 1 shows the mean values of software metric $\gamma_{review/design}$ for three project groups.

Table 1: Comparison of ratio of review effort

	Vending system (1992-1994)	Checking system (1992-1996)	Retail system (1995-1996)
$\gamma_{review/design}$	8.9%	11.8%	20.6%

In Table 1, the project group Retail System seems to succeed in following the guideline G_1 faithfully. However, the project group Checking System seems to fail to follow the guideline G_1. Thus, we execute the test of statistical hypothesis(t-test) with 5% level of significance to $\gamma_{review/design}$'s of Retail System and Checking System. As a result, we can prove that there exists a significant difference between them.

We discuss the characteristics of organizations to show the reason why there exists a distinct difference between Retail System and Checking System. The project group Checking System started in 1992 and the members of the organization for Checking System had already established their own ways for software development when the SEPG started the review process improvement. Thus it is hard for most of them to change the process instantly according to the guidelines

specified by the SEPG group. On the contrary, the project group Retail System started after the SEPG had determined the guidelines. Thus the members of the organization for Retail System tend to accept the guideline without difficulties.

As mentioned above, we should discuss the properties of organizations rather than that of individual projects. Thus for convenience we define a set of faithful projects as a faithful project group. According to this definition, we refer

Retail System (1995–1996): faithful project group

Checking System (1992–1996): unfaithful project group

in the following. To tell the truth, Checking System includes four faithful projects. But, it is also clear the organization that developed Checking System failed to follow the guideline $\boldsymbol{G_1}$. Thus we interpret that the organization happened to have the result $\gamma \geq 15\%$ for some projects in Checking System, and assume that Checking System is an unfaithful project group. For convenience, we also refer

Vending System (1992–1994): unfaithful project group

since this project group contains only unfaithful projects.

3 Effectiveness Analysis

3.1 Software metrics and assertion

At first, we explain the fundamental metrics related to the quality collected in the development process shown in Figure 1. The metric F_{review} represents the total number of faults detected by all review activities in design phase. The metric F_{test} represents the total number of faults detected in the debug & test phase. During six months after the code shipping, the SEPG provides the monitoring phase and collects all data concerning the failures detected by customers, as shown in Figure 1. We call these failures post-released failures, and use the metric $F_{monitor}$ to represent the number of post-released failures.

Using these data collected from the projects, we define software metrics ρ and χ to evaluate the quality of software product.

(1) Ratio of detected faults (ρ's: %)

In order to evaluate the ratio of detected faults in a specific phase to all the faults detected, we define three metrics as follows:

$$\rho_{review/total} = \frac{F_{review}}{F_{review} + F_{test} + F_{monitor}} \times 100$$

$$\rho_{test/total} = \frac{F_{review}}{F_{review} + F_{test} + F_{monitor}} \times 100$$

$$\rho_{monitor/total} = \frac{F_{review}}{F_{review} + F_{test} + F_{monitor}} \times 100$$

The metric $\rho_{review/total}$ stands for the ratio of faults detected in the review to all faults detected, and $\rho_{test/total}$ stands for the ratio of faults detected in the debug & test phases. Finally, $\rho_{monitor/total}$ stands for the ratio of post-release failure detected in the monitor phase.

(2) Seriousness of failure (χ : level)

For each post-release failure, the maintenance operator and the SEPG jointly decide the seriousness χ. The values of seriousness χ are as follows: *destructive*, *confusing* and *mild*. The level χ =*destructive* means that the failure can lead to the system down. Thus the failure must be removed immediately. Then the level χ =*confusing* means that only a part of system may be down by the failure and other parts may keep working. Thus it should be removed immediately if possible. Finally the level χ =*mild* means that the failure never affects the essential part of the system, and thus it may be negligible for a while.

We evaluate the effectiveness of review process improvement quantitatively. Thus, we investigate the changes in the number of detected faults and the changes in the quality of the final product. They are the indirect effects, but are the most essentially expected effects of the review process improvement.

We try to prove the following assertions $\boldsymbol{A_2}$ and $\boldsymbol{A_3}$ by statistical analysis:

- $\boldsymbol{A_2}$ The number of faults detected by the review increases in each project of the faithful project group. Similarly, the number of faults detected in the debug & test phase decreases.
- $\boldsymbol{A_3}$ As a result of the review process improvement, the quality of the final code is also improved.

3.2 Ratio of detected faults ρ

Now we investigate the effect of review process improvement concerning the assertion $\boldsymbol{A_2}$. Table 2 shows the mean values of software metrics $\rho_{review/total}$(the ratio of faults detected in review phase to all the faults detected), $\rho_{test/total}$ (the ratio of faults detected in debug & test phase to all the faults detected) and $\rho_{monitor/total}$ (the ratio of post-release failures to all the faults). In Table 2, we classify the projects into faithful project group (that is, Retail System) and unfaithful project group(Vending System and Checking System).

By the test of statistical hypothesis with 5% level of significance, all of the values $\rho_{review/total}$ and $\rho_{test/total}$ confirmed the significant difference between the faithful project group and the unfaithful project group. Thus, we can say the correctness of the assertion $\boldsymbol{A_2}$ is confirmed by statistical analysis.

However, the values of $\rho_{monitor/total}$ shown in Table 2 seems to indicate that there are no difference between the ratio of

Table 2: Comparison of ratio of detected faults ρ

	Faithful project group	Unfaithful project group
$\rho_{review/total}$	78.4%	38.8%
$\rho_{test/total}$	21.1%	60.7%
$\rho_{monitor/total}$	0.5%	0.5%

post-release failures in faithful and unfaithful projects. Then we have to perform another analysis to investigate the post-release failure.

3.3 Post-release failure

As mentioned before, for each post-release failure the SEPG or the maintenance operator decide the seriousness χ and assign its value to the failure. Table 3 summarizes the total number of post-release failures and the distributions of χ's assigned to post-release failures. Because the values of these metrics are confidential, we cannot publish the values themselves in this paper. Thus Table 3 shows only the relative values by assuming all the values for the unfaithful project group to be one.

Table 3: Comparison of post-release failures

		Faithful project group	Unfaithful project group
$F_{monitor}$ / # of projects		0.55	1
χ	destructive	0.44	1
	confusing	0.50	1
	mild	0	1

Since the original values of these metrics are so small, we cannot apply the statistical analysis. However, in Table 3 we can observe some interesting properties. The total number of post-release failures($F_{monitor}$/# of projects) of the faithful project group is smaller than that of the unfaithful project group. Especially, the number of the failures with $\chi =$*destructive* is smaller drastically. Thus we can guess the correctness of the assertion $\boldsymbol{A_3}$.

4 Conclusion

We have analyzed the effectiveness of the review process improvement activities by the SEPG during these six years in a certain company. According to the guidelines determined by the SEPG, we have investigated the ratio γ of the review effort to the total effort for design and coding activities. As a result we found that a newly started project group followed the guidelines faithfully. Similarly, we have investigated the ratio ρ of faults detected in review to the total number of detected faults. The result showed that the ratio ρ is improved drastically in a faithful project group. Finally, we have confirmed the number of post-release failures during six months after code release is also decreased by the SEPG's process improvement activities. Furthermore, we confirmed the number of critical failures are also decreased.

Acknowledgements

Authors would like to thank Dr. Shinji Kusumoto of Osaka University and Mr. Naoki Niihara, Mr. Yasunari Takagi and Mr. Keishi Sakamoto of OMRON Corporation for their discussions and advice to our analysis in this paper.

REFERENCES

[1] L. C. Briand, K. E. Emam, B. Freimut and O. Laitenberger: "Quantitative evaluation of capture-recapture models to control software inspections," Proc. of 8th Int'l Symposium on Software Reliability Engineering, pp.234–244, 1997.

[2] R. G. Ebenau and S. H. Strauss: *Software Inspection Process*, McGraw-Hill, 1993.

[3] M. E. Fagan: "Advances in software inspections," IEEE Trans. on Software Engineering, Vol.12, No.7, pp.744–751, 1986.

[4] W. S. Humphrey: *Managing the Software Process*, Addison Wesley, Reading, MA, 1989.

[5] O. Mizuno, T. Kikuno, K. Inagaki, Y. Takagi and K. Sakamoto: "Analyzing effects of cost estimation accuracy on quality and productivity," Proc. of 20th Int'l Conference on Software Engineering(ICSE'98), pp.410–419, 1998.

[6] M. C. Paulk, C. V. Weber, S. M. Garcia, M. B. Chrissis and M. Bush: "Key practice of the capability maturity model, version 1.1," Technical Report CMU/SEI-93-TR-25, Software Engineering Institute, 1993.

[7] Y. Takagi, T. Tanaka, N. Niihara, K. Sakamoto, S. Kusumoto and T. Kikuno: "Analysis of review's effectiveness based on software metrics," Proc. of 6th Int'l Symposium on Software Reliability Engineering(ISSRE'95), pp.34–39, 1995.