# ある企業におけるソフトウェアプロセス改善の効果に対する統計的分析

水野 修†，二木 俊樹‡，新原 直樹‡，高木 徳生‡，菊野 亨†

†大阪大学 大学院基礎工学研究科 情報数理系専攻
〒560-8531 大阪府豊中市待兼山町 1-3
Phone: 06-850-6567　　Fax: 06-850-6569
e-mail: o-mizuno@ics.es.osaka-u.ac.jp
‡オムロン株式会社

あらまし: 本研究ではある企業におけるレビュープロセス改善活動の効果について統計的分析を行う．この企業では1995年よりソフトウェアプロセスグループ(SEPG)を中心にこの改善活動が進められてきており，分析に当たっては23件のプロジェクトから収集されたデータを用いる．まず，各プロジェクトにおいてレビュー作業工数の全体工数に対する比率に注目して，開発組織がSEPGからのプロセス改善の指示をどの程度忠実に実施できたかを分析する．次に，レビュープロセス改善の品質への影響について調べる．その結果，プロセス改善を忠実に実施できた組織とそうでない組織の間ではレビュー作業で検出するフォールト数に有意水準5%の検定で差があることが示された．更に，出荷後のいわゆるフィールド品質についても相当の改善が見られることが確認できた．
キーワード: ソフトウェアプロセス，レビュープロセス改善，ソフトウェア品質，統計的分析

# Effectiveness Analysis of Review Process Improvement for Embedded Software System Development at Certain Company

Osamu Mizuno†, Toshiki Niki‡, Naoki Niihara‡, Yasunari Takagi‡and Tohru Kikuno†

†Department of Informatics and Mathematical Sciences,
Graduate School of Engineering Science, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
Phone: +81-6-850-6567　　Fax: +81-6-850-6569
e-mail : o-mizuno@ics.es.osaka-u.ac.jp
‡OMRON Corporation

Abstract: In this paper, we analyze the effectiveness of actual review process improvement activities conducted by the software engineering process group(SEPG) in a certain company that develops embedded software systems.

The analysis consists of two steps: investigating the ratio $\gamma$ of the efforts for review on the total efforts for design and coding(Step 1), and clarifying the effects caused by the review process improvement(Step 2). The Step 1 classifies projects into two categories: *faithful* and *unfaithful* projects, where $\gamma \geq 15\%$ holds for faithful projects. Then Step 2 proves statistically that the number of faults detected during the review process in faithful projects is significantly greater than that of unfaithful projects. Additionally, it also shows that the number of serious failures in faithful projects is slightly smaller compared with unfaithful projects.

Keywords: software process, review process improvement, quality of product, statistical analysis

# 1 Introduction

In order to achieve high quality and productivity of the software development, a lot of general knowledge or experiences such as "no silver bullet[3]" and "death march project[18]" have been presented. The process improvement[7] is well known as one of the most attractive and practical methodologies. Concerning the process improvement, theoretical investigations on the formal framework[4, 15] as well as applications to actual organizations [8, 17] have been performed. Especially, the process improvement of software review has been considered as a cost-effective way to improve quality of software[5]. Actually, several quantitative evaluations of software review process have proved its validity using simulation results[2, 9].

In this paper, we analyze the effectiveness of review process improvement activities during six years in a certain company that develops embedded software systems. In the company, software engineering process group(SEPG) was established in 1992, then the activities for constructing well-formed project plans were introduced in 1993, and finally the activities for improving review process have been conducted since 1995. Concerning the case studies of software process improvement using generalized stochastic Petri-net model and the statistical analysis of the relationship between the well-formed project plan and the final product quality, we have already summarized some reports[11, 16, 17]. But the discussions on review process improvement are not yet done, and so we try to analyze empirically the effectiveness of review process improvement using actual twenty three project data from 1992 to 1996. In this study, we assume review includes both the document review in the design activity and the code review in the coding activity.

The objective of this study is to investigate carefully the long-term changes in the software development process which are introduced by the review process improvement in the company. In this paper, we try to investigate the following two changes on the projects: changes in review effort and changes in software quality. At first, we trace direct effects on the development process caused by the review process improvement, such as the change of review efforts actually spent for each project, and checks if the process improvement is truly incorporate into the actual development. Next, we trace indirect effects caused by the review process improvement, such as the changes in the number of faults detected in the review and test phases, and verify if the quality of software products is essentially improved.

The rest of the paper is organized as follows: In Section 2, we present target projects and process model. In Section 3 we explain the process improvement performed from 1992 to 1998 at a certain company. In Section 4, we show the outline of collected data and clarify the objective of our study by presenting three assertions. In Section 5, we show the result of evaluations with respect to review effort and software quality. Finally, we conclude this paper with a brief summary and some future research works in Section 6.

# 2 Development at Certain Company

## 2.1 Target projects

In order to derive as general results as possible, we select 23 projects such that development effort of each project is larger than 15 person-months. These selected projects are classified into three kinds of categories based on their products: Vending System, Checking System, and Retail System as follows:

Vending System: The developments of user-friendly vending machines, that accept cash or prepaid cards, for the railroad system. All of the selected five projects began in the period from 1992 to 1994. Typically, it takes from 15.3 to 37.2 person-months for developing the vending machine with 9.4 to 38.8 K-step.

Checking System: The developments of checking machines for the fare in the train stations and airports. All of the selected twelve projects began in the period from 1992 to 1996. Typically, it takes 16.3 to 62.3 person-months for developing the checking machine with 7.2 to 123.1 Kstep.

Retail System: The developments of retail systems allowing consumers to use credit and prepaid cards. All of the projects began in the period from 1995 to 1996. Typically, it takes 15.0 to 29.2 person-months for developing retail system with 3.5 to 20.5 Kstep.

The one important point for later analysis is that the Vending System and Checking System are developed by the same organization in the company. On the other hand, the Retail System is developed by another organization.

Figure 1 summarizes the categories of 23 projects and period of their developments. Please note that activities of the SEPG for improving review process have been conducted since 1995, to be described in Section 3.

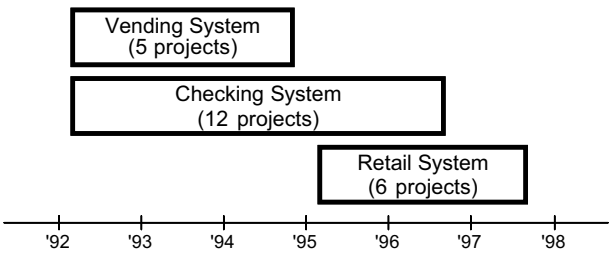


Figure 1: Target projects

## 2.2 Process model

Figure 2 shows the software development process model adopted in the company. The process model shown in

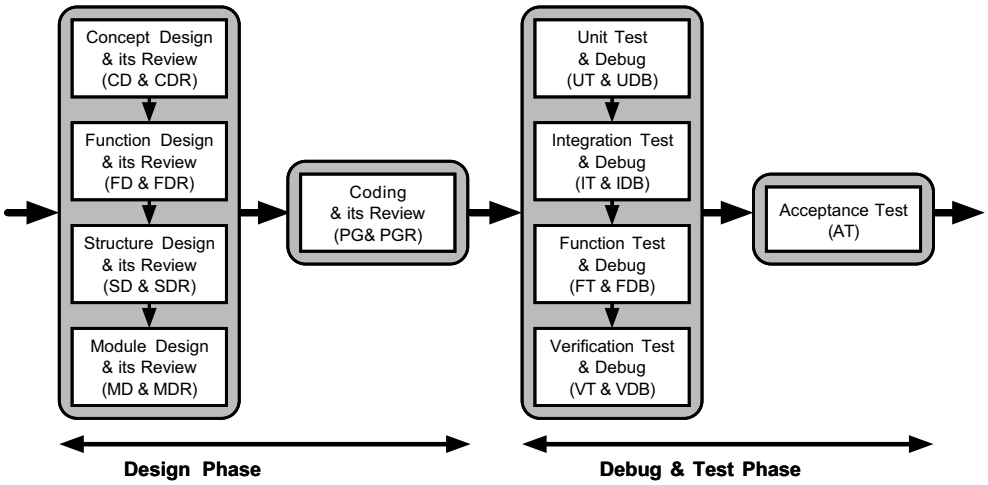


Figure 2: Process model

Figure 2 is a standard waterfall model. Strictly speaking, some irregular control flows (such as backwards flow to previous activity or concurrent executions between previous and current activities) do rarely happen. But these are not explicitly described in Figure 2.

In our discussion, we classify the development process into two successive phases: design phase and debug & test phase, as shown in Figure 2. One important characteristic of the design phase is that the review activity is introduced after each design activity and coding activity. The design phase is divided into five stages: Concept Design, Function Design, Structure Design, Module Design, Coding and their corresponding Reviews. On the other hand, the debug & test phase consists of the repetition of a pair of test and debug activities and is divided into four stages: Unit Test, Integration Test, Function Test, Verification Test and their corresponding Debugs. At the end of the debug & test phase, there is Acceptance Test, where final test before code release is performed.

# 3 Process Improvement

## 3.1 Overview

In the certain company to be investigated in this paper, the software engineering process group(SEPG) was established in 1992. Since then the SEPG 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. Figure 3 shows a brief summary of the process improvement activities at the company.

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 the 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 with level of significance 5% in [11].

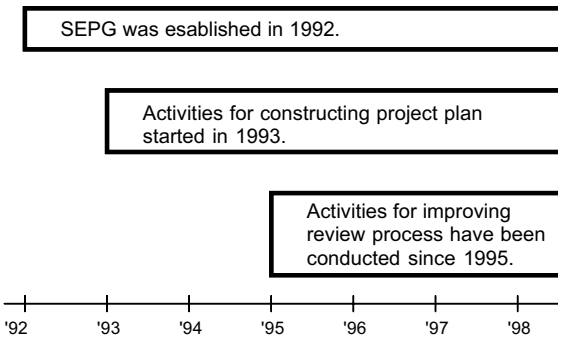


Figure 3: Process improvement activities

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, it is believed that the number of faults detected in the review activity become 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.

### 3.2 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 guideline or numerical target value for the document review[5, 7]. On the other hands, 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[16].

Based on the analysis result of the past project data, the SEPG has derived the following guidelines $\boldsymbol{G1}$–$\boldsymbol{G6}$ for the review activities:

$\boldsymbol{G1}$ At least the 15% of the total efforts for design and coding activities should be assigned to reviews(the document review and the coding review).

$\boldsymbol{G2}$ Reviewers must report the progress using the standard review form at regular intervals.

$\boldsymbol{G3}$ 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[13]).

$\boldsymbol{G4}$ The coding review should be performed by two or three persons, including one person who develops the code.

$\boldsymbol{G5}$ The review coverage rate for the code review should be about 200 lines of code per hour.

$\boldsymbol{G6}$ All of new codes and changed codes should be reviewed. (Concerning reuse of old codes, reviews are not necessary required.)

Among them, $\boldsymbol{G1}$ and $\boldsymbol{G2}$ are general requirements for reviews(including both the document review and the code review), $\boldsymbol{G3}$ is specific for the design review, and $\boldsymbol{G4}$, $\boldsymbol{G5}$ and $\boldsymbol{G6}$ are only for the code review.

Before the review improvement began, the same activities as the guideline $\boldsymbol{G2}$, $\boldsymbol{G3}$ and $\boldsymbol{G4}$ were required to be performed in the review. However, since the guidelines were not yet established, the review was 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 from 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 are truly 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.

## 4 Objective of Our Study

### 4.1 Software metrics

Here, we explain software metrics: $\gamma$, $\rho$ and $\chi$ to be measured in each project. Figure 4 shows a simplified process model and a part of fundamental data set collected at each phase of development.

From the design phase, the SEPG collects data of six fundamental metrics: $E_{design}$, $E_{review}$, $E_{coding}$, $E_{c.review}$ for the efforts spent and $F_{review}$, $F_{c.review}$ for the number of detected faults. 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. Similarly, the metrics $E_{coding}$ and $E_{c.review}$ represent the effort spent for coding and coding review, respectively. Next the metric $F_{review}$ represents the total number of faults detected by all review activities in design phase, and the metric $F_{c.review}$ represents the number of faults detected by coding review activity.

From the debug & test phase, the SEPG collects data of two fundamental metrics: $E_{test}$ and $F_{test}$. The metric $E_{test}$ represents the total efforts spent for all activities in the debug & test phase. Next, 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 failure detected by customers, as shown in Figure 4. 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 three kinds of software metrics $\gamma$, $\rho$ and $\chi$ to analyze and evaluate the software development process.

(1) Ratio of review effort ($\gamma$'s: %)

In order to evaluate the amount of efforts, we define two metrics as follows:

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

$$\gamma_{c.review/coding} = \frac{E_{c.review}}{E_{coding} + E_{c.review}} \times 100$$

The metric $\gamma_{review/design}$ evaluates the ratio of all review effort to design and coding efforts, and $\gamma_{c.review/coding}$ evaluates the ratio of code review effort to coding effort.

(2) Ratio of detected faults ($\rho$'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:

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

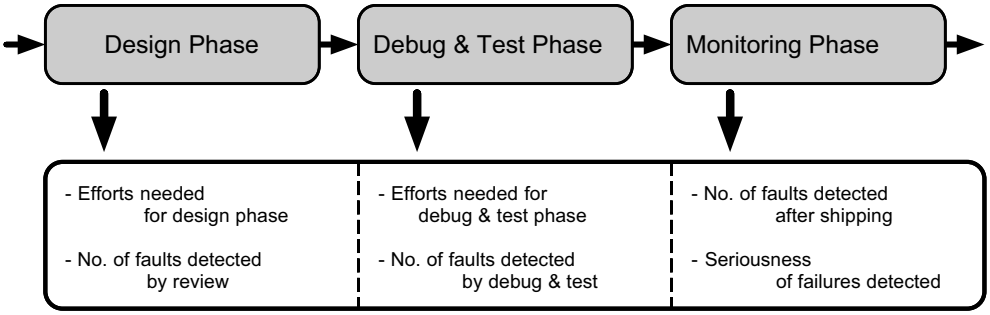


Figure 4: Fundamental data set

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

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

The metric $\rho_{review/total}$ evaluates the ratio of faults detected in the review to all faults detected, and $\rho_{c.review/total}$ evaluates the ratio of faults detected in the code review to all faults detected. Finally, $\rho_{test/total}$ evaluate the ratio of faults detected in the debug & test phase to all faults detected.

(3) Seriousness of the failure ($\chi$ : level)

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

## 4.2 Assertions

The objective of our study is to investigate the effectiveness of review process improvement conducted by the SEPG. We try to apply statistical analysis to the data collected in the actual projects to clarify the changes in the review effort and the changes in the software quality. The analysis consists of two step: investigating the review effort ratio (Step 1) and investigating the software quality (Step 2).

(1) Review efforts (Step 1 of analysis)

Here we try to confirm that the review improvement activities are accepted in the development team. 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 activity. As mentioned in subsection 3.2, the guideline $\boldsymbol{G_1}$ 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 affect by the guideline $\boldsymbol{G_1}$. Based on these considerations, we try to prove the following assertion $\boldsymbol{A_1}$ by statistical analysis:

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

Intuitively speaking, the assertion $\boldsymbol{A_1}$ means that the effort for review activity increases as the result of the SEPG's guidance. We call a project that satisfies the guideline $\boldsymbol{G_1}$ (that is, the 15% of the total efforts on design and coding activities is assigned to the review activity) a *faithful* project group. In the evaluation of process improvement we should discuss the properties of organization rather than that of individual projects. Then intuitively speaking, we define a set of faithful project as a faithful project group.

(2) Software quality (Step 2 of analysis)

Here we try to 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 affect, but are the most essentially expected affect of the review process improvement.

In Step 2 of analysis, 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 and test phase decreases.

$\boldsymbol{A_3}$ As the result of the review process improvement, the quality of the final code is also improved.

Table 1: Comparison of ratio of faithful projects

| | Vending System (1992-1994) | Checking System (1992-1994) | Checking System (1995-1996) | Retail System (1995-1996) |
|---|---|---|---|---|
| No. of projects | 5 | 7 | 5 | 6 |
| No. of faithful projects | 0 | 2 | 2 | 6 |

Table 2: 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% |
| $\gamma_{c.review/coding}$ | 6.6% | 10.5% | 21.6% |

Intuitively speaking, the assertion $\boldsymbol{A_2}$ means that by increasing the ratio of review effort to the total efforts on design and coding activities, we can change the software process essentially. That is, we can increase the number of faults detected in the review, and at the same time we can reduce the number of faults detected in the debug and test activities.

The assertion $\boldsymbol{A_3}$ means that even the improving the quality of the final product will be partly attained by the review process improvement. In the analysis in Section 5, we apply data on the post-release failure that are collected during six months after shipping, to evaluate the quality of the final product.

# 5 Effectiveness Analysis

## 5.1 Ratio of review effort $\gamma$

At first, concerning the assertion $\boldsymbol{A_1}$, 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 consider project group rather than an individual project. As explained already in subsection 2.1, there exist three project groups: Vending System, Checking System and Retail System.

As mentioned already in subsection 3.2, the guideline $\boldsymbol{G_1}$ 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. Then Table 1 shows the number of faithful projects in each project group. In Table 1, in order to investigate Checking System (1992-1996) in detail, we divide it into Checking System (1992–1994) and Checking System (1995–1996) based on the year 1995 when the SEPG started the review process improvement. (Please note that the review process improvement activities started in 1995, and thus Vending System is out of scope.) Next, Tables 2 and 3 show the mean values of software metrics $\gamma_{review/design}$ and $\gamma_{c.review/coding}$ (the ratio of code review effort to coding effort) for three project groups.

From Tables 1 and 2, the project group Retail System seems to succeed to follow the guideline $\boldsymbol{G_1}$ faithfully. However, the project group Checking System seems to fail to follow the guideline $\boldsymbol{G_1}$. Thus, we execute the test of statistical hypothesis with 5% level of significance to $\gamma_{review/design}$'s of Retail System and Checking System. As the result, we can prove that there exists a significant difference between them. Similarly, for $\gamma_{c.review/coding}$, we can also prove that there exists a significant difference between that of Retail System and Checking System.

We discuss the characteristics of organizations to show the reason why there exists a big 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 generally hard for 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 beyond all question.

Next, from Table 3 we can see the mean values of $\gamma_{review/design}$ and $\gamma_{c.review/coding}$ in 1995–1996 become greater than that in 1992–1994. Thus, the review improvement seems to be accepted by the organization little by little. However, from the test of statistical hypothesis with 5% level of significance, we cannot see a significant difference for $\gamma_{review/design}$ and $\gamma_{c.review/coding}$ between

Table 3: Comparison of review effort in Checking System

| | Checking System (1992-1994) | Checking System (1995-1996) |
|---|---|---|
| $\gamma_{review/design}$ | 11.0% | 12.9% |
| $\gamma_{c.review/coding}$ | 9.6% | 11.8% |

1992–1994 and 1995–1996.

As already mentioned, 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. Clearly by this definition, 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 four projects in Checking System, and take a view point 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.

### 5.2 Ratio of detected faults $\rho$

Now we investigate the effect of review process improvement concerning the assertion $\boldsymbol{A_2}$. Table 4 shows the mean values of software metrics $\rho_{review/total}$(the ratio of faults detected in review phase to all the faults detected), $\rho_{c.review/total}$(the ratio of faults detected in code review to all the faults detected) and $\rho_{test/total}$ (the ratio of faults detected in debug & test phase to all the faults detected). In Table 4, we classify the projects into faithful project group (that is, Retail System) and unfaithful project (Vending System and Checking System).

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

### 5.3 Post-release failure

As mentioned before, for each post-release failure the SEPG or the maintenance operator decide the seriousness $\chi$ and assign its value to the failure. Table 5 summarizes the total number of post-release failures and the distributions of $\chi$'s assigned to post-release failures. However,

Table 4: Comparison of ratio of detected faults $\rho$

| | Faithful project group | Unfaithful project group |
|---|---|---|
| $\rho_{review/total}$ | 78.4% | 38.8% |
| $\rho_{c.review/total}$ | 21.7% | 12.7% |
| $\rho_{test/total}$ | 21.1% | 60.7% |

the values of these metrics are confidential, we cannot publish the values themselves in this paper. Thus Table 5 shows only the relative values by assuming all the values for unfaithful project group to be one, and the symbol $*$ denotes its representation.

Since the original value of these metrics are so small, we cannot apply the statistical analysis. But from Table 5 we can observe some interesting properties. The total number of post-release failure ($F_{monitor}$/# of projects) of faithful project group is smaller than that of unfaithful project group. Especially, the number of the failures with $\chi =$*destructive* is smaller drastically. Thus we can guess the correctness of the assertion $\boldsymbol{A_3}$.

Table 5: Comparison of post-release failures

| | | Faithful project group | Unfaithful project group |
|---|---|---|---|
| $F_{monitor}$/ # of projects | | 0.55* | 1* |
| $\chi$ | *destructive* | 0.44* | 1* |
| | *confusing* | 0.50* | 1* |
| | *mild* | 0* | 1* |

## 6 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 $\gamma$ 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 $\rho$ of faults detected in review to the total number of detected faults. The result showed that the ratio $\rho$ is improved drastically in a faithful project group. Finally, we have confirmed the number of post-release failure during six months after code release is also decreased by the SEPG's process improvement activities.

## References


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

[2] L.C. Briand, K.E. Emam, O. Laitenberger and T. Fussbroich: "Using simulation to build inspection efficiency benchmarks for development projects," Proc. 20th International Conference on Software Engineering(ICSE'98), pp.340–349, 1998.

[3] F.P. Brooks Jr.: "The Mythical Man-Month," Addison Wesley, 1975.

[4] A. Cimitile and G. Visaggio : "A formalism for structured planning of a software project," International Journal of Software Engineering and Knowledge Engineering, Vol.4, No.2, pp.277–300, 1994.

[5] R.G. Ebenau and S.H. Strauss: "Software inspection process," McGraw-Hill, 1993.

[6] N.E. Fenton and S.L. Pfleeger: "Software Metrics : A Rigorous & Practical Approach," PWS Publishing, 1997.

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

[8] W.S. Humphrey, T. Snyder and R. Willis: "Software process improvement at Hughes Aircraft," IEEE Software, Vol.8, No.4, pp.11–23, 1991.

[9] S. Kusumoto: "Quantitative evaluation of software reviews and testing process," PhD. Dissertation, Osaka University, 1993.

[10] S. Kusumoto, O. Mizuno, Y. Hirayama, T. Kikuno, Y. Takagi and K. Sakamoto: "A new project simulator based on generalized stochastic Petri-Net," Proc. 19th International Conference on Software Engineering(ICSE'97), pp.293–303, 1997.

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

[12] K.H. Möller and D.J. Paulish: "Software Metrics : A Practitioner's Guide to Improved Product Development," IEEE Press (Chapman & Hall Computing), 1993.

[13] 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.

[14] A.A. Porter, H.P. Siy, C.A. Toman and L.G. Votta: "An experiment to assess the cost-benefits of code inspections in large scale software development," IEEE Transactions on Software Engineering, Vol. 23, No. 6, pp.329–346, 1997

[15] R.M. Podorozhny and L.J. Osterweil: "The criticality of modeling formalisms in software design method comparison," Proc. 19th International Conference on Software Engineering(ICSE'97), pp.303–313, 1997.

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

[17] T. Tanaka, K. Sakamoto, S. Kusumoto and T. Kikuno: "Improvement of software process by process visualization and benefit estimation," Proc. 17th International Conference on Software Engineering(ICSE'95), pp.123–132, 1995.

[18] E. Yourdon: "Death March : The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects," Prentice Hall Computer Books, 1997.