PAPER

# A Model to Explain Quality Improvement Effect of Peer Reviews

**Mutsumi KOMURO**[†a)], ***Member and*** **Norihisa KOMODA**[††], ***Nonmember***

**SUMMARY** Through the analysis of Rayleigh model, an explanatory model for the quality effect of peer reviews is constructed. The review activities are evaluated by the defect removal rate at each phase. We made hypotheses on how these measurements are related to the product quality. These hypotheses are verified through regression analysis of actual project data, and concrete calculation formulae are obtained as a model. Making use of the mechanism to construct this model, we can develop a method for making concrete review plan and setting objective values to manage on-going review activities.

***key words:*** *process model, peer review, quality, improvement effect*

## 1. Introduction

It is important to know the quantitative effect of process improvement activities in order to correctly and effectively implement the improvements. This is particularly true when we conduct process improvement movement at the organizational level. For example, at the maturity level 4 of CMMI®* the organization is supposed to establish a process performance model which is used for (i) establishing organization and project objectives and verifying their reasonableness, (ii) determining whether the project is on track to meeting its objectives, (iii) analyzing and/or predicting impact, benefit, and ROI when evaluating and/or selecting process improvement activities, (iv) evaluating effects of a change on process performance [1]. One of the most popular processes chosen for improvement in level 4 is peer review [2].

Peer review is known to be a simple, effective, and inexpensive way of detecting and removing defects from software work products. Peer review is conducted not only on source code but also on various documents. There have been many proposals for peer review methods such as Fagan inspection [3], active design reviews [4], n-fold inspections [5], and two-person inspections [6]. It is known that peer reviews in upper-stream phases have greater values than those in lower-steam phases [7]. The work products reviewed in upper-stream phases are mainly specification and design documents. There are also performance studies of peer review to investigate what kind of factors affect the results of peer reviews, such as productivity or defect density [8]. It is important to know exactly how peer reviews affect the product quality in order to navigate the process improvements to the right direction. This is especially true for peer reviews on documents in upper-stream phases. However, only a few studies [9], [10] have been done to establish a model to quantitatively explain its effect on product quality in terms of peer review performance.

Rayleigh model is known to be useful to describe various software lifecycle patterns. It can also be applied to describe the defect removal pattern of software projects [9]. However, the examples described in [9] are from very large size projects following rigid waterfall lifecycle phases which are not very popular these days. One major issue when applying the Rayleigh model would be how to represent the time variable. Although we only have discrete ordering of time represented by lifecycle phases in software development, the Rayleigh model assumes the existence of a continuous time variable. In [9] a simple arithmetic progression are allocated for waterfall development phases. However, this approach does not make much sense when the projects do not follow rigid waterfall phases.

PSP^SM (Personal Software Process^SM)** [10] contains many useful practices for software development. Empirical data are given to show that some measurements have negative correlation with defect density at testing phases. These measures include what Humphrey calls 'yield' which is defined as the percentage of product defects that are removed in each phase, and the 'appraisal/failure cost ratio' which is a ratio of review time to compile and test time. However, no quantitative model is given to explain how much these measures affect the defect density.

In [11], Nakano, et al. investigated the relationship between the quality of source code and the performance data consisting of defect density and review efficiency of source code reviews. Here the quality of the source code is evaluated by defect density at testing phases and review efficiency is defined as (review size)/(man hour spent for the review). They classified peer review data into subgroups according to review efficiency values and observed that too high defect density leads to bad quality of the source code within the same subgroup. However, they did not give any further refined model, e.g., a formula to compute the defect density at the testing phase in terms of the review efficiency

Manuscript received March 5, 2009.
Manuscript revised August 11, 2009.

[†]The author is with Engineering Support Center, Hitachi Software Engineering, Tokyo, 140–0002 Japan.
[††]The author is with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan.
a) E-mail: komuro@hitachisoft.jp

DOI: 10.1587/transinf.E93.D.43

*CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
**PSP is a service mark of Carnegie Mellon University.

and the defect density at the source code review. They only dealt with source code review. Peer reviews in upper-stream phases such as reviews for specification documents were not considered.

In this paper we propose a model to explain quality improvement effect of peer reviews. We represent the intensity of peer reviews by defect removal rate and relate it to the product quality represented by defect density at testing phase. In order to do that, we introduce an intermediate variable measuring the ratio of remaining defects at testing phases to the total number of defects throughout the development phases. Unlike the Rayleigh model, our model does not assume the existence of a proper time variable and is more flexible with the choice of lifecycle model. This mechanism can be used to establish a plan and objective values for review activities to ensure product quality given in terms of defect density in testing phase. This plan can be further refined to define concrete review activities, if performance analysis results such as difference among various peer review methods are available.

## 2. Related Results

### 2.1 Rayleigh Model and its Properties

The cumulative distribution function (CDF) and probability density function (PDF) of Rayleigh distribution are given as follows.

CDF: $F(t) = 1 - e^{-\frac{t^2}{c^2}}$,

PDF: $f(t) = \frac{2t}{c^2} e^{-\frac{t^2}{c^2}}$,

where $c$ is the scale parameter.

The Rayleigh distribution is a special case of the Weibul distribution, which is originally defined in the reliability engineering. In the context of reliability engineering, the complement of $F(t)$ is more important and called the reliability function [12]. Let us denote it as $R(t) = 1 - F(t)$. The ratio of $f(t)$ and $R(t)$ is another useful measure called failure rate in reliability engineering. We denote it as $d(t) = f(t)/R(t)$. In reliability engineering the parameter '$t$' represent the time and the behaviors of $R(t)$ and $d(t)$ are studied.

The Rayleigh distribution is also used in software engineering to describe various software lifecycle patterns [13]. In the software development, defects are detected throughout the development phases. This detection is mainly conducted by peer reviews in the upper-stream phases and by testing in the lower-stream phases. If we plot the number of defects found in each phase sequentially, we usually observe one peak in some middle phase. Intuitively this would suggest the Rayleigh curve could be used as a model to describe the defect removal pattern. There are several empirical studies [9], [14], [15] and even a computer program was developed to estimate the number of latent defects. These results are summarized in [9]. It would be crucial to choose proper time variable $t$ to apply Rayleigh model in this way. [9] suggests to use a simple arithmetic progression of 0.5, 1.5, 2.5, 3.5, 4.5, and 5.5 for consecutive development phases, i.e., the first phase is represented by 0.5, the second phase is represented by 1.5, and so on.

In this application to the defect count, we assume that the probability density function $f(t)$ represent the probability of detection of a defect at time $t$ and that the reliability function $R(t)$ represents the ratio of remaining defects at the time point $t$. It should be noted that $R(t)$ at two time points $t_0$ and $t_1$ ($t_0 < t_1$) are related as

$$R(t_1) = e^{-t_1^2/c^2} = (e^{-t_0^2/c^2})^{t_1^2/t_0^2} = R(t_0)^{t_1^2/t_0^2} \quad (1)$$

Substituting the explicit formulae of CDF and PDF into the definition of $d(t)$, we obtain

$$d(t) = 2t/c^2. \quad (2)$$

When we fix the time to be $t_0$, we can solve this formula to get the value of $1/c^2$ and substitute it in $R(t_0)$ to obtain

$$R(t_0) = e^{-d(t_0)t_0/2} \quad (3)$$

From the assumptions about $f(t)$ and $R(t)$ mentioned above, this $d(t_0) = f(t_0)/R(t_0)$ represents the ratio of defects detected at time $t_0$ to the total number of defects remained at $t_0$. A similar measure is defined in [10] and called 'yield.' Strictly speaking, our $d(t)$ is different from Humphrey's yield which does not count the defects injected after the time $t$. We call $d(t)$ as defect removal rate at $t$.

Equation (3) implies that if we raise the defect removal rate at some time $t_0$, it will decrease the remaining defects exponentially. Furthermore, combining Eqs. (1) and (3), we get

$$R(t_1) = e^{-d(t_0)t_1^2/2t_0} \quad (4)$$

Therefore $d(t_0)$ also affects exponentially on $R(t_1)$ at a later time point $t_1$.

### 2.2 General Results in Reliability Engineering

These Eqs. (1) and (4) are special cases of general relationship between the reliability function $R(t)$ and the density function $f(t)$ in reliability engineering [12].

Since $R(t)$ is a monotone decreasing continuous function with $1 > R(t) > 0$ for $t > 0$, it is obvious that $R(t_1)$ can be written as $R(t_0)^k$ by some positive constant $k$ for $t_1 > t_0 > 0$. In the case of Rayleigh distribution, Eq. (1) gives an explicit expression $t_1^2/t_0^2$ for this constant $k$. In particular, this constant $k$ is determined only by the two time points $t_1$ and $t_0$ independent of the parameter $c$.

Since $f(t)$ is equal to $dF(t)/dt = -dR(t)/dt$, $d(t) = f(t)/R(t)$ can be written as

$$d(t) = -R'(t)/R(t) = -d(\log(R(t)))/dt.$$

It follows that

$$\log(R(t_0)) = -\int_0^{t_0} d(t)dt$$

In the case of Rayleigh model, Eq. (3) states that the integral on the right hand side can be replaced by a single value $t_0 d(t_0)/2$.

### 2.3 Limitation of Rayleigh Model

As we mentioned, [9] uses a simple arithmetic progression as the values of the time variable for development phases. This choice might be right for large size projects with rigid waterfall development phases, as described in [9].

However, recent software development projects have different characteristics. Although the waterfall lifecycle model is still popular, most middle or small size projects do not exactly follow these development phases in our company. Sometimes phases are merged, skipped, or even concurrently executed. It is clear that a simple arithmetic progression does not work for these projects. Actually we studies 33 project data to see whether Eq. (2) can be satisfied with a simple arithmetic progression. The result shows wider variances and far from satisfying Eq. (2). Therefore, Rayleigh model can not be simply applied to these projects.

## 3. Our Hypotheses and Verification

### 3.1 Statements of Hypotheses

As mentioned above, the Rayleigh distribution curve does not always fit well with the numbers of defects found in recent software projects. However, if the Rayleigh model is so successful for large size projects as described in [9], some of its properties may be still valid for recent projects.

It is well known that the quality of development process affects significantly on the product quality [16], [17]. Furthermore, we often experience that the quality achieved in upper-stream phases of development has substantial effects on that of the later phases [18]. Equations (3) and (4) can be regarded as quantitative expressions of this fact.

Although we do not assume the Rayleigh model, we would like to make hypotheses that the discrete equivalent of Eq. (4) holds. In order to state these hypotheses, first let us reformulate the measures $f(t)$, $R(t)$, $d(t)$ in discrete setting. Since we do not have a continuous time variable, we will use a variable $p$ representing a phase of development. That is, $p$ ranges over a finite set of symbols representing development phases, e.g., {BS, FS, DS, P, T} or {I0, I1, I2, UT, CT, ST}. Thus, we will use measures $f(p)$, $R(p)$, or $d(p)$ instead of $f(t)$, $R(t)$, or $d(t)$.

We assume that we have consecutive development phases $p_j$ $(1 \leq j \leq n)$ with testing phases located at last parts of the sequence. For any phase $p_j$, we denote the number of defects detected at the phase $p_j$ as $f(p_j)$. Let $R(p_j)$ be the proportion of the defects detected at phases later than $p_j$ to all the defects found in a development. That is,

$$R(p_j) = \sum_{k>j} f(p_k) \Big/ \sum_{1\leq k\leq n} f(p_k)$$

Let $d(p_j)$ be the ratio of $f(p_j)$ to the sum of all the defects at the phase $p_j$ or later, i.e.,

$$d(p_j) = f(p_j) \Big/ \sum_{k\geq j} f(p_k)$$

Motivated by the formula (4) for the Rayleigh distribution, we make the following hypotheses.

**Stronger Hypothesis A** *Let $p_j$ and $p_k$ be two development phases with $j \leq k$. Then we have a relation of the form*

$$R(p_k) = e^{-C_{jk}d(p_j)}$$

*with some positive constant $C_{jk}$.*

The important special case is when $p_k$ is the phase just before the testing. Let us denote this phase as $P$.

**Hypothesis A** *Let $p_j$ be a development phase before testing. Then we have a relation of the form*

$$R(P) = e^{-C_j d(p_j)}$$

*with some positive constant $C_j$.*

The first hypothesis is stronger statement, because it relates any two phases $p_j$ and $p_k$. Since our main concern is about the product quality, we only deal with the Hypothesis A not the Stronger Hypothesis A in the rest of this paper.

We would like to look into the relationship between $R(P)$ and the product quality. Intuitively a lower $R(P)$ value means better process quality and hence would lead to better product quality. But first we need to clarify what we mean by the product quality.

Let $DD_T$ be the defect density at testing phases. Although the product quality is a complex concept consisting of reliability, usability, functionality, security, safety, performance, compatibility, and so forth, $DD_T$ is the most popular measure to evaluate the product quality. This is because the reliability is an important prerequisite for realizing high product quality. The density of defects found after the development would be a better measure for reliability than $DD_T$. However, $DD_T$ is much easier to collect values and these two measures are known to have significant positive correlation [9]. Thus we can use $DD_T$ to represent the product quality.

By its definition $R(P)$ is equal to the number of defects at testing phases divided by all the defects throughout the development phases. The definition of $DD_T$ is very similar. It is defined as the number of defects at testing phases divided by the development size. Thus, the numerators are the same and the difference lies in the denominators. Note that under the Hypothesis A the value of $d(p)$ can be determined by $R(P)$ for any phase $p$ before $P$. Hence, under the Hypothesis A if we are given the value of $R(P)$ and the sequence of development phases, then we can determine the shape of the distribution of the numbers of defects over these phases.

With the situation stated above, the only way to vary a value of $R(P)$ is to multiply some constant to these distributed numbers. For example, if the value of $R(P)$ becomes three times larger, then the number of defects at each phase becomes three times larger, too.

It follows that if we could control this multiplying constant, $R(P)$ should be proportional to $DD_T$. Since this multiplying constant affects all the development phases, it is clearly independent of the performance of peer review at

each phase and may be determined by some global or external factor such as overall difficulty of the system under development, sufficiency of experiences of project manager and developers, clarity and feasibility of the user requirement, or availability of related systems with critical dependencies. This is our second hypothesis.

**Hypothesis B** *If the global or external conditions of the project are the same, then $R(P)$ is proportional to $DD_T$.*

With Hypotheses A and B we can explain the effect of peer review on the product quality in terms of $d(p)$ of each development phase $p$. Once these hypotheses are verified, we can decide how many defects should be removed by peer review activities in each phase, which enables projects to plan, conduct, and evaluate the peer review activities in a quantitative manner. This is especially useful for peer reviews in upper-stream phases, because they are most remote from the final product and it is not always clear how they affect the product quality quantitatively.

### 3.2 Verification of Hypotheses

#### 3.2.1 Used Data and Some Background Information

In this section we will investigate the Hypotheses to see whether they hold for actual project data. We investigated performance data from thirty three completed projects collected throughout our company. These projects conducted peer reviews at various phases. As described in [19], we already conducted organizational analysis on peer review performance data, identified best practices of peer reviews, and delivered training courses on these best practices. We are also using control charts to monitor and control peer review activities. Therefore these performance data is fairly stable. In particular, defects counted by each project are standardized so as not to include minor defects like typographical errors.

Since we have a set of standard processes at company level, these projects used lifecyle model and review process derived from those of the standard processes with suitable tailoring [19]. Some projects use iterative lifecycle model, but the Waterfall lifecyle model is still popular. Our standard process defines development phases for these lifecyle models. In this paper we use the following symbols for the development phases in the Waterfall lifecycle model.

BS: basic specification,
FS: functional specification,
DS: detailed specification,
P: programming,
T: overall testing phase, i.e., union of unit testing, combination testing, and system testing.

In order to verify Hypotheses A and B, we need to deal with the numbers of defects on programming and testing phases, which depend on the programming language the project uses. In the 33 projects we selected, Java was the most popular programming language. We selected 17 projects which employed Waterfall lifecycle model and developed in Java out of 33 projects. This choice of programming language naturally leads to the exclusion of very large systems in financial or public service business domains where COBOL is still the major language. The selected projects typically develop Web or network systems for various customers including manufacturing logistics, and telecommunication industries.

#### 3.2.2 Verification of Hypothesis A

In order to see whether Hypothesis A holds, we investigate the linear relationship with zero intercept between $\log(R(P))$ and $d(p_j)$, i.e., a relationship of the form $\log(R(P)) = -C_j d(p_j)$ with some positive constant $C_j$.

We performed linear regression analyses for $p_j$ = BS, FS, DS, and P with the data from 17 projects mentioned above.

The scatterplots with regression lines for FS is given in Fig. 1 and Table 1 shows the results of these regression tests.

The results of regression analyses are summarized in Table 1. These R-square values are pretty large and p-values are very small. Judging from these results, Hypothesis A seems to be certainly valid.

#### 3.2.3 Verification of Hypothesis B

First we modify Hypothesis B which is stated in terms of $R(P)$ and $DD_T$. The measure $DD_T$ has some drawback, especially when we deal with statistical data. Although defect density is a measure normalized by development size, in general it still has negative correlation with the development size [20].

This is because $DD_T$ treats data of a large system in

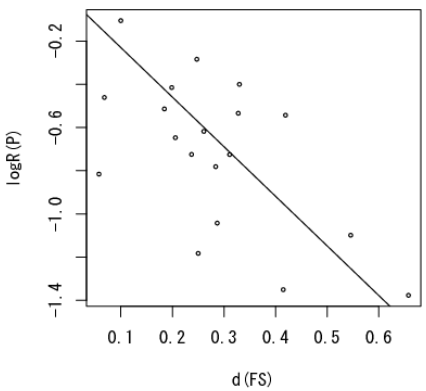


**Fig. 1** Scatterplot of $d(FS)$ and $\log(R(P))$.

**Table 1** Results of regression analyses.

| Phase | Regression line | $R^2$ | p-value |
|---|---|---|---|
| BS | $\log(R(P)) = -9.93d(BS)$ | 0.762 | 0.00097 |
| FS | $\log(R(P)) = -2.23d(FS)$ | 0.793 | 7.35e-7 |
| DS | $\log(R(P)) = -2.62d(DS)$ | 0.806 | 1.33e-5 |
| P | $\log(R(P)) = -2.91d(P)$ | 0.875 | 1.25e-8 |

the same way as that of a small system. Since the former is generally more stable and reliable than the latter, the former should be considered as more important than the latter. Let $prj$ be a variable to represent a project. For each $prj$, let us denote the number of defects found at testing phases, the development size, and the defect density at testing phases by $D(prj)$, $S(prj)$, and $DD_T(prj)$ respectively.

Since the development processes are standardized, we assume that the defects are (caused and) detected at certain probability. We can estimate this probability by computing the average $\Sigma_{prj}D(prj)/\Sigma_{prj}S(prj)$. Let us denote this value by $q$. Since each project has the development size $S(prj)$, the defect density $DD_T(prj)$ is associated with the standard deviation $\sqrt{q(1-q)/S(prj)}$. This implies that $DD_T(prj)$ becomes $\sqrt{k}$ times more reliable as $S(prj)$ becomes $k$ times larger. We would like to renormalize the deviation $DD_T(prj)-q$ as $(DD_T(prj)-q)\sqrt{S(prj)}$. We call this new measure as the normalized defect density and denote it by $\overline{DD_T(prj)}$ or simply by $\overline{DD_T}$ when the dependency on the project is understood from the context:

$$\overline{DD_T(prj))} = (DD_T(prj) - q)\sqrt{S(prj)}.$$

We reformulate the Hypothesis B to the following modified Hypothesis B which we are going to verify by linear regression.

**Modified Hypothesis B**
If the global or external conditions of the project are the same, there is a linear relationship of the form

$$\overline{DD_T} = aR(P) + b$$

with suitable constants $a$ and $b$.

Note that unlike Hypothesis B the intercept $b$ of the modified Hypothesis B may not be zero, because we are dealing with the normalized measure. That is, zero defects in testing phases does not necessary mean $\overline{DD_T} = 0$. In fact $\overline{DD_T} = 0$ means $DD_T = q$.

First let us look at the scatterplot of $\overline{DD_T}$ and $R(P)$ shown in Fig. 2. Performing linear regression analysis with single explanatory variable $R(P)$, we got the regression line plotted and the following values: $R^2$: 0.43, p-value: 0.0080.

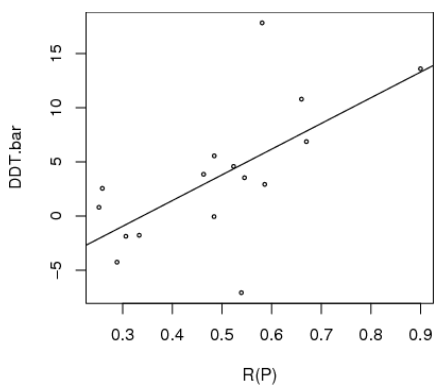


**Fig. 2** Scatterplot of $\overline{DD_T}$ and $R(P)$.

The p-value is sufficiently low, but the R-square value is not very high. This result implies that $R(P)$ has significant correlation with $\overline{DD_T}$, but the explanatory variable $R(P)$ does not solely explain the behavior of the response variable $\overline{DD_T}$. There may be other factors to affect $\overline{DD_T}$.

Examining each project data, we found that each outlier has certain characteristics such as the technical difficulty of the development is low/high, the project manger had little/much experience of similar development, or development environment was good/bad. We made evaluation list consisting of about 30 items and interviewed the projects. This list was derived from risk evaluation sheet traditionally used within our company. These 30 items were Yes/No or High/Middle/Low type questions based on the criteria for risk analysis.

We represented the answers as logical variables and performed linear regression analyses with dummy variables. We selected variables with statistically significant effect with 5% significance level and further chose the following three variables based on AIC (Akaike's Information Criteria) [21]. (1) pm: representing whether the chief senior developer has sufficient experience of developing similar system or not, (2) dp: representing whether the developing process is based on the standard process of our company or derived from the customer's processes, (3) wp: representing whether requirements on size and quality of work products is unambiguous. Although these dummy variables do not necessary have one-to-one correspondence with the characteristics of the outliers mentioned above, (1) definitely has a correspondence and the other two seem to have certain relationship with the remaining characteristics.

The resulting regression formula is given as follows.

$$\overline{DD_T} = -6.7 + 20.8R(P) + 22.3pm - 12.2dp + 10.9wp$$

with multiple $R^2$: 0.76, adjusted $R^2$: 0.69, p-value: 0.0002. Since p-value is sufficiently small, the correlation is certainly significant. The multiple and adjusted R-square values suggest that the selected explanatory variables reasonably explain the variance of the response variable. This result is not too bad, but we have an impression that we could improve this result by stratifying the data into projects of different business domains. Current data set is too small to do this.

## 4. An Application of Our Approach —— Objective Value and Review Plan

The explanatory model described in the previous section gives quantitative evaluation results for peer review activities. From the practical point of view, this is useful to navigate the process improvement for peer reviews to the right direction. It would be more useful if we can predict the final improvement effect of review activity while it is being conducted.

The explanatory model does not have this ability, because it relies on the value of the defect removal rate $d(p_j)$

at phase $p_j$ which can not be calculated until the completion of the project. However, we can use the mechanism described in the previous section to develop review plan and set objective values for review activities which can be used to control the on-going review activities.

When we make a plan, we estimate the development size and set an objective value of $DD_T$. The project information is evaluated as a part of risk analysis, including the technical difficulty and experience of similar system development. Then, by using the model in a reverse way, we can decide what kind of value or value range the defect removal rate at each phase should take. It follows that we can decide how many defects we should remove at each phase. This helps make concrete review plan, including what kind of review method is used, how many review meetings should be held, etc.

We performed the regression analysis in a converse order. That is, we treat $R(P)$ as a response variable and $\overline{DD_T}$ as an explanatory variable. As was mentioned in the statement of Modified Hypothesis B, $R(P)$ and $\overline{DD_T}$ have linear correlation if the global or external conditions of the projects are the same. In the verification of the Modified Hypothesis B, we chose three dummy variables to represent the global or external conditions. Since we are investigating a different functional relationship by exchanging the roles of response and explanatory variables, we need different variables to represent the global or external conditions. We investigated the evaluation list of the project information and chose the following three variables based on AIC, again: (1) pm: representing whether the chief senior developer has sufficient experience of developing similar system or not, (2) hd: representing whether development requires a concurrent hardware development or not, (3) ac: representing whether the system requires complex architecture design or not.

Note that the first variable is the same as the one before, while the other two are different. This is because response and explanatory variables have asymmetric roles in regression analysis. When we exchange $x$ and $y$, the linear equation obtained by minimizing the square residuals will generally change. It follows that set of outlier points will not remain the same. Hence the selection of explanatory variables can be changed.

The resulting regression formula is given as follows.

$$R(P) = 0.22 + 0.020\overline{DD_T} + 0.17pm + 0.31hd + 0.10ac \tag{5}$$

with multiple $R^2$: 0.71, adjusted $R^2$: 0.60, p-value: 0.0053. Since p-value is sufficiently small, the correlation is certainly significant. The multiple and adjusted R-square values are not too bad. Again, we have an impression that we could improve this result by stratifying the data into projects of different business domains. We also performed a series of regression analyses with $d(p)$ for each development phase $p$ as the response variable and $R(P)$ as single explanatory variable. The result is summarized in Table 2.

From these results, we can compute the expected values of the number of defects in each phase: First, we get the value of $R(P)$ by Eq. (5) and compute each $d(p_j)$ by the Table 2. Based on the estimated development size and the objective value of the defect density at the testing phase, we can compute an expected value of the total number of defects in testing phase. That is, we get an estimated value of $f(T)$. From this information we can compute the number of defects at each phase consecutively. For example, for the programming phase $P$ we get the following formula by the definition of $d(P)$.

$$d(P) = f(P)/(f(P) + f(T))$$

It follows that

$$f(P) = d(P)f(T)/(1 - d(P)).$$

Since all the terms in the right hand side are already computed, we get the value of $f(P)$. Repeating this procedure for all the phases, we can obtain the value of $f(p)$ for $p = P, DS, FS, BS$.

**Table 2** Results of regression analyses.

| Phase | Regression line | $R^2$ | p-value |
|---|---|---|---|
| BS | $d(BS) = -0.075\log(R(P))$ | 0.762 | 0.0021 |
| FS | $d(FS) = -0.33\log(R(P))$ | 0.766 | 2.06e-7 |
| DS | $d(DS) = -0.30\log(R(P))$ | 0.786 | 1.07e-5 |
| P | $d(P)) = -0.31\log(R(P))$ | 0.860 | 4.11e-9 |

These expected values are very useful for projects, because they give concrete target values to manage the projects. We have already established various performance baselines for several measures of peer review, including review rate (review size per review hour), and defect density (number of defects per review size). These baselines are classified according to the key features of peer reviews, including review method, involvement of experts, number of review members, and system architecture. Once the expected values are calculated, the project can plan what kind of peer reviews they will conduct based on the expected value, the baseline information, and the project situation.

Usually, projects select several different review features according to their needs and available resources. Typically, there are three categories. The first one is the peer reviews for core or important part of the system for which rigorous review method such as Fagan inspection is chosen. Review participants must include key persons like domain experts. The second one is review for the part developed by novice developers. Popular peer reviews for this part is walkthrough, which can be used as training for novice developers [22]. The reviews for the rest of the system are classified as the third category. Light review method may be chosen for this category, including two-person reviews [6] and personal reviews. These three categories are given priorities in this order.

Based on the size of work products to be reviewed and the baseline data of defect density for the review feature selected, number of defects detected for each category is estimated. Then the baseline data of review rate is used to estimate how many hours each part of the system should

be reviewed. Review schedule is created and the feasibility is examined. The estimated numbers of defects for each category are summed up and checked whether the objective value is satisfied. If there is any problem or conflict, that should be resolved. The category with higher priority has a precedence to be reallocated necessary resources.

## 5. Conclusion and Future Work

Through the analysis of Rayleigh model, we constructed an explanatory model for the quality effect of peer reviews. The review activities at each phase $p$ are evaluated by the defect removal rate $d(p)$. We made hypotheses on how these measurements are related to the product quality represented by the defect density at testing phase.

We verified these hypotheses through regression analysis of actual project data and obtained concrete calculation formulae. Making use of the mechanism to construct this model, we can develop a method for making concrete review plan and setting objective values for review activities to manage on-going review activities.

As future work, we are going to verify the usefulness of the model through actual application to the project activities, including planning, monitoring & control.

### References

[1] M.B. Chrissis, M. Konrad, and S. Shrum, CMMI Guidelines for Process Integration and Product Improvement, Addison-Wesley, 2003.
[2] M.C. Paulk and M.B. Chrissis, "The 2001 high maturity workshop," Tech. Rep. 2001-SR-014, CMU, 2001.
[3] M. Fagan, "Design and code inspections to reduce errors in program development," IBM Syst. J., vol.15, no.3, pp.182–211, 1976.
[4] D.L. Parnas and D.M. Weiss, "Active design reviews: Principles and practices," J. Syst. Softw., vol.7, no.4, pp.259–265, 1987.
[5] G.M. Schneider, J. Martin, and W.T. Tsai, "An experimental study of fault detection in user requirements documents," ACM Trans. Softw. Eng. Methodol., vol.1, no.2, pp.188–204, 1992.
[6] D.B. Bisant and J.R. Lyle, "A two-person inspection method to improve programming productivity," IEEE Trans. Softw. Eng., vol.15, no.10, pp.1294–1304, 1989.
[7] K.E. Wiegers, Peer Reviews in Software, Addison-Wesley, 2002.
[8] A. Porter, H. Siy, A. Mockus, and L. Votta, "Understanding the sources of variation in software inspections," ACM Trans. Softw. Eng. Methodol., vol.7, no.1, pp.41–79, 1998.
[9] S.H. Kan, Metrics and Models in Software Quality Engineering, Addison-Wesley, 2003.
[10] W.S. Humphrey, PSP(SM) A Self-Improvement Process for Software Engineers, Addison-Wesley, 2005.
[11] Y. Nakano, O. Mizuno, T. Kikuno, Y. Anan, and M. Tanaka, "Analysis on impact of defect density and effiency of coding review to software qualtiy," SEC Journal, vol.2, no.8, pp.10–17, 2006.
[12] R. Ramakumar, Engineering Reliability, Prentice-Hall, 1993.
[13] L.H. Putnam, "A general empirical solution to the macro software sizing and estimation problem," IEEE Trans. Softw. Eng., vol.4, no.4, pp.345–361, 1978.
[14] M. Trachtenberg, "Discovering how to estimate software reiability," RCA Engineer, vol.27, no.1, pp.53–57, 1982.
[15] J. John and E. Gaffney, "On predicting software related performance of large-scale systems," Int. CMG Conference, pp.20–23, 1984.
[16] R.S. Kenett and E.R. Baker, Software Process Quality, Marcel Dekker, 1999.
[17] K. Ishikawa, Introduction to Quality Management, JUSE Press, 1989.
[18] K. Yasuda and T. Nara, Introduction to Software Quality Assurance, JUSE Press, 2008.
[19] M. Komuro, K. Otokozawa, and Y. Kimura, "Improvement of peer review process based on quantitative effect analysis of actual performance of projects," SEC Journal, vol.1, no.4, pp.6–15, 2005.
[20] IPA SEC, White Paper on Software Development Data, Nikkei BP., 2008.
[21] H. Akaike, S. Amari, G. Kitagawa, S. Kabashima, and H. Simodaira, Akaike Information Criterion, Kyoritsu Shuppan, 2007.
[22] D.A. Wheeler, B. Brykczynski, J. Reginaald, and N. Meeson, Software Peer Reviews, R. Thayer ed., IEEE Computer Society Press, 1997.

**Mutsumi Komuro** received his B.S. and M.S. degrees in mathematics from the University of Tokyo. He also received M.S. degree in computer science from Stanford University. Currently he is working as a Chief Consultant in Hitachi Software Eng. Co., Ltd. His research interests include quantitative process improvement, empirical software engineering, and project management for software development.

**Norihisa Komoda** is a professor of the Department of Multimedia Engineering, Graduate School of Information Science and Technologies, Osaka University since 2002. He received the B. Eng. and M. Eng. degrees in electrical engineering and the Dr. Eng. from Osaka University in 1972, 1974, and 1982, respectively. He was with Systems Development Laboratory, Hitachi Ltd. from 1974 to 1991. In 1991, he moved to Faculty of Eng., Osaka University. His research interests include business information systems, electronic commerce systems, and knowledge information processing.