Construction of Quality Prediction Model based on Peer Review Performance Data

M. Komuro¹ and N. Komoda²
¹Engineering Support Center, Hitachi Solutions, Ltd., Tokyo, Japan
²Graduate School of Information Science and Technology, Osaka University, Osaka, Japan

Abstract—Firstly, a model is constructed which describes the relationship between the number of defects detected by tests and those detected by peer reviews. Secondly, a method is given to transform this model into a prediction model. Finally, the quality prediction model obtained by this method is validated through actual project data. The resulting model can predict the number of defects at testing phase with ±18% ~ ±24% relative error.

Keywords: Process Improvement, Peer Review, Process Performance, Quantitative Model, Quality Prediction

1. Introduction

It is important to know the quantitative effect of process improvement activities in order to correctly and effectively implement the improvements. Furthermore, if we can predict the effect of the improvement from the performance data of currently on-going processes, we can avoid the possible future trouble or more effectively perform the processes. For example, in CMMI (Capability Maturity Model Integration)¹ which is a standard of process improvement model this kind of mechanism is called process performance model and is considered as an important characteristic of high maturity[1].

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. It is known that peer review has both the quality improvement effect and cost reduction effect [2][3].

When a project has some quality problem, often it is not realized until the testing phase. However, it is too late or very costly to improve the quality in the testing phase. If we have some mechanism to predict the quality at the testing phase from the peer review performance data at the upper stream phases, it would be useful to prevent this kind of failures.

In this paper we propose a method to construct a model to predict the number of defects at the testing phase from the peer review performance data. We validate the resulting model through real world project data.

2. Effect Evaluation Model

We assume that the Waterfall lifecycle model is employed and sequential lifecycle phases are used for software development projects. Let us denote these phases as $p_j$ ($1 \leq j \leq n$) and further assume that the last several phases are allocated for testing. Since we are primarily interested in defect removal by peer reviews and we will not use each of these testing phases separately, we treat these several testing phases as one single phase denoted by a phase symbol $T$. We also assume that the phase just before the phase $T$ is allocated for programming and denote it as a phase symbol $P$ as illustrated in Fig. 1.

Let $f(p_j)$ be the number of defects detected at the j-th phase $p_j$. When $p_j = T$, $f(T)$ is the number of defects of the program detected by testing. When the phase $p_j$ is equal or earlier than $P$, $f(p_j)$ is the number of defects detected by peer reviews. The work product under review is source code in $p_j = P$ and development documents such as specification and design documents in the earlier phases.

Let us define $d(p_j)$ by $d(p_j) = f(p_j)/\Sigma_{k \geq j} f(p_k)$ and call it as the defect removal rate at phase $p_j$. Intuitively $d(p_j)$ represents how intensively the peer review is conducted at phase $p_j$, comparing with the performance of later phases.

If we conduct peer reviews intensively at some phase $p_j$ so as to increase the defect removal rate, then the quality of the products later than $p_j$ would be improved. Therefore it is natural to expect the defect density at testing phase will decrease. The following 'Effect Evaluation Model' represents this expectation in a quantitative way.

(Effect Evaluation Model)

For any phase $p_j$ before the testing we have

$$\log (\text{test.defect}) = a_1 \log (\text{size}) + a_2 d(p_j) + \beta_1 + \ldots + \beta_n + c,$$  

(1)

where test.defect is the number of defects detected at the testing phase, size is the size (KLOC) of the developed program, $d(p_j)$ is the defect removal ratio at phase $p_j$, $a_1, a_2, c$ are constants, and $\beta_1, \ldots, \beta_n$ are terms representing the effect of factors other than peer review performance. It should be noted that defects are mainly removed through
peer reviews at phase \( p_j \), which is before the testing and that \( d(p_j) \) represents the quality enhancement effect of these peer review activities.

This model assumes linear relationship between \( \log(test\_defect) \) and \( \log(size) \) like COCOMO quality estimation model[4]. In our previous paper [2] it is shown that \( d(p_j) \) affects exponentially on the defect density at testing phase. Although this Effect Evaluation Model deals with \( test\_defect \) directly rather than the defect density at the testing phase, we still assume that \( d(p_j) \) affects exponentially on \( test\_defect \). Hence, \( d(p_j) \) appears directly on the right hand side without \( \log \) transformation. This can be viewed as a representation of a practical observation [5][6] that performance of peer reviews in upper stream phases strongly affects the quality in lower stream phases.

We validate this model by regression analysis using actual project data which are collected from a business division in a software development company. We refer to this division as ‘the target organization’ in the rest of this paper. These projects in the target organization build Web based software systems using Java language. We will use 35 data out of these projects excluding those which have missing records on some of development phases because they order suppliers to build a part of the development system.

The target organization has been improving their peer review processes [3][7]. A standard set of checklist items for peer review is established and used by projects through tailoring. This selection of checklist items defines what kind of defects should be detected at peer reviews.

We apply Generalized Linear model[8][9] to this validation. In our case, \( test\_defect \) is not a continuous but a discrete random variable derived from counts of defects. For continuous variables, the distribution of the residuals is often the normal distribution, for which classical linear model taking the least square sum of residuals to measure the goodness of the fit instead of the least square sum of the residuals. It is also possible to specify link functions to transform the response variables in Generalized Linear models and the canonical link function is determined for each residual distribution. For example, the canonical link function for Poisson distribution is the logarithm \( \log(x) \) [9].

Since the response variable is already transformed by the logarithm in the Effect Evaluation Model, we take the logarithm for the link function.

Two kinds of variables are used for the \( \beta \) term.

(a) Risk evaluation results

These are dummy variables with values in the set \{0, 1\} obtained from the evaluation results of risk analysis. In the target organization the risks are evaluated and given a numerical value between 1 and 5 as defined in standard risk evaluation sheet. We select about 30 check items expected to be related to the product quality. If the evaluation is 4 or 5, the risk is judged to be high and specific action is taken to monitor or mitigate the risk. We introduce a dummy variable for each item representing whether the evaluation is 4 or 5.

\[ \log(base\_size+1) \] : logarithm of base system size

Often the software system to develop has some base system, i.e., the previously developed system on which the current development is based. Let us denote the KLOC of the base system as base\_size. Qualitatively this variable represents the amount of experience on the developing system. As with the variable size of development size, we take the logarithm of this variable. We add 1 to base\_size to prevent it from becoming negative infinity.

We first select the variables which have 5% statistical significance and then add or delete the variable checking whether the addition of the variable decreases residual deviance significantly. Since the addition of a dummy variable is logically equivalent to the introduction of a case classification, we carefully avoid adding too many dummy variables compared to the number of data points.

In the target organization the following symbols are used to represent development phases \( p_j \). BS: a phase to develop basic specification, FS: a phase to develop functional specification, DS: a phase to develop detailed Design, P: a phase for programming. These development phases are deployed in this order right before the testing phase. We conduct the linear regression analysis for each phase and get the result shown in Table 1. The line with the phase symbol BS + FS + DS shows the result of regression analysis considering the three phases BS, DS, and FS as one development phase in which document reviews are conducted. We use quasi-Poisson distribution instead of Poisson distribution, because we observed over dispersion. That is, the residual deviance is much greater than the one expected from the degrees of freedom.

<table>
<thead>
<tr>
<th>( p_j )</th>
<th>( \beta ) term</th>
<th>p-value</th>
<th>coeff.</th>
<th>pseudo ( R^2 )</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS</td>
<td>LB</td>
<td>0.046</td>
<td>-2.43</td>
<td>96.7%</td>
<td>10</td>
</tr>
<tr>
<td>FS</td>
<td>CO, DE</td>
<td>6.6 x 10^{-8}</td>
<td>-2.13</td>
<td>96.6%</td>
<td>34</td>
</tr>
<tr>
<td>DS</td>
<td>LE, AC</td>
<td>0.037</td>
<td>-0.83</td>
<td>91.8%</td>
<td>30</td>
</tr>
<tr>
<td>BS+ FS+ DS</td>
<td>DE</td>
<td>0.038</td>
<td>-0.92</td>
<td>89.8%</td>
<td>35</td>
</tr>
<tr>
<td>P</td>
<td>RC, DD</td>
<td>3.4 x 10^{-9}</td>
<td>-1.86</td>
<td>92.6%</td>
<td>34</td>
</tr>
</tbody>
</table>

The explanatory variables selected for \( \beta \) terms are the following. LB: logarithm of KLOC of base system, i.e., \( \log(base\_size+1) \), CO, DE, AC, RC, and DD are dummy variables representing the risk about customer organization, development environment, determination of system
architecture, requirements documentation, and delivery date respectively. We limit the number of dummy variables no greater than 2 for each regression equation.

The column under ‘#’ shows the number of data points i.e., projects used for each regression. The reasons why the number of projects differs for each phase is because some projects skip some phases. Especially in the target organization there are many projects which skip the earliest phase BS and start from FS. There are two cases for this as follows. (1) Projects developing the system with a base system may not need to execute BS again. (2) The customer or another company performs the upper stream phases and a project in the target organization develops the system based on the requirements made in these upper stream phases.

The p-value shown in Table 1 is that of \(d(p_j)\) and the pseudo \(R^2\) is McFadden’s pseudo \(R^2\). It shows the percentage of determination by log likelihood instead of the percentage of determination by sums of squares in the usual linear regression model. In every phase the value of pseudo \(R^2\) is about 90%, which can be an evidence that the Effect Evaluation Model is valid. On the other hand, the occurrence of overdispersion suggests that there are some missing explanatory variables. However, we do not add dummy variables any more, because the number of data points are limited to at most 35 as shown above.

We also conduct simple linear regression analysis for each explanatory variable with a response variable \(\log(test: \text{defect})\). As a result, it is found that \(\log(size)\) has the largest pseudo \(R^2\) value. The result of this regression is as follows. p-value: smaller than 2 × 10^{-16}, pseudo \(R^2\): 83.0%, regression coefficient: 0.92, standard error of regression coefficient: 0.06.

This result shows that the prime factor to determine the behaviour of \(\log(test: \text{defect})\) is \(\log(size)\). It follows that we can get a approximated value of \(test: \text{defect}\) by a formula of the form \(test: \text{defect} = A \times size^{0.92}\) or more simply a formula of the form \(test: \text{defect} = A \times size\) can be used, judging from the value of the standard error.

3. Construction and Validation of Quality Prediction Model

In this section we will construct a model to predict the number of defects in testing phase based on the Effect Evaluation Model described in the previous section.

3.1 Iterative Prediction Model

Our idea to construct prediction model is to consider the Effect Evaluation Model as an equation for \(test: \text{defect}\) and solve it by the iteration method. Since the Effect Evaluation Model is established through regression analysis, it contains an error term. However, in order to explain our idea clearly, we ignore the error term for a while.

For the sake of simplicity, let us first consider when \(p_j = P\). Then the defect removal rate at the phase \(P\) is given by \(d(P) = f(P)/(f(P) + test: \text{defect})\). Let us consider the right hand side as a function of \(test: \text{defect}\) and denote it as \(d_P(test: \text{defect})\). That is, \(d_P(x) = f(P)/(f(P) + x)\). Note that \(d_P(test: \text{defect}) = d(P)\) holds. Then the following proposition holds.

(Construction of Iterative Prediction Model)
Assume that the equation of the Effect Evaluation Model \(test: \text{defect} = size^{a_1} \exp(c + a_2 d_P(t_{d0}) + \beta_1 + \cdots + \beta_n)\) (2) holds without an error term and that the absolute value of the coefficient \(a_2\) is smaller than 4.

If \(t_{d0}\) is a good enough approximating value for \(test: \text{defect}\), then

\[
size^{a_1} \exp(c + a_2 d_P(t_{d0}) + \beta_1 + \cdots + \beta_n)
\]

will give a better approximation of \(test: \text{defect}\) than \(t_{d0}\).

[Proof] Let \(\varepsilon\) be the difference between \(t_{d0}\) and \(test: \text{defect}\), i.e., \(t_{d0} = test: \text{defect} + \varepsilon\) and assume that \(\varepsilon\) is small enough.

Let us denote \(size^{a_1} \exp(c + a_2 d_P(x) + \beta_1 + \cdots + \beta_n)\) as \(g(x)\). Then \(g(test: \text{defect}) = test: \text{defect}\) holds by the assumption. It follows that

\[
g(t_{d0}) - test: \text{defect} = g(test: \text{defect} + \varepsilon) - g(test: \text{defect})
\]

Expanding the first term in Taylor series, the last expression is equal to the following, \(g'(test: \text{defect})\varepsilon + \text{(higher terms)}\), where \(g'(x)\) denotes the first derivative of the function \(g(x)\).

Ignoring the higher terms, we thus obtain

\[
|g(t_{d0}) - test: \text{defect}| = |g'(test: \text{defect})\varepsilon|
\]

By a straightforward calculation we get

\[
|g'(test: \text{defect})| = |a_2| \times \frac{|f(P)\text{test: defect}|}{|f(P) + f(P)\text{test: defect}|^2}
\]

However, by the well-known inequality of arithmetic and geometric means, we have

\[
\sqrt{test: \text{defect} f(P) \leq \frac{test: \text{defect} f(P)}{2}}
\]

Thus we obtain

\[
|g(t_{d0}) - test: \text{defect}| \leq |\frac{a_2}{2}| < |\varepsilon|
\]

because of the assumption \(|a_2| < 4\). This inequality shows that \(g(t_{d0})\) is a better approximation of \(test: \text{defect}\) than \(t_{d0}\). [QED]

As shown in the ‘coeff.’ column in Table 1 the condition \(|a_2| < 4\) in the proposition holds for every development phase for the projects in the target organization. Therefore this construction can be applied to the data in the target organization.

The intuitive reason why \(size^{a_1} \exp(c + a_2 d_P(t_{d0}) + \beta_1 + \cdots + \beta_n)\) gives a better approximation than \(t_{d0}\) is because the latter explicitly uses the information about the peer review performance. While the above proof is given under the assumption that the equation of the Effect Evaluation Model holds without an error term, we can expect \(size^{a_1} \exp(c + a_2 d_P(t_{d0}) + \beta_1 + \cdots + \beta_n)\) is still a better approximation if \(t_{d0}\) does not use the information about peer review performance.

Under the assumption that the equation of the Effect Evaluation Model holds without an error term, we could repeat the
above procedure to compute the value of \texttt{test.defect} more and more precisely. We call the prediction model obtained by this iteration as \textit{Iterative Prediction Model}.

### 3.2 Quality Prediction Model

Now we know that the Iterative Prediction Model gives an approximating value of \texttt{test.defect}. However, the iteration of regression analysis could be very complicated and the resulting model would be hard to maintain. In this section we will show a way to ‘transform’ the Iterative Prediction Model into another model involving only one iteration. In order to do this, first we actually construct a model implementing the idea of ‘iteration’ and investigate its property.

1) Selection of initial solution

We choose an initial solution to be proportional to the development size (KLOC), i.e., a solution of the form \( \text{test.defect} = k \cdot \text{size} \), where \( k \) is the regression coefficient. As mentioned in the end of section 2, \( \log(\text{test.defect}) \) and \( \log(\text{size}) \) have a strong positive correlation with regression coefficient close to 1. This would be one of the natural selections.

2) Second approximation

Using the above initial solution, we compute the right hand side of the Effect Evaluation Model.

\[
\begin{align*}
& a_1 \log(\text{size}) + a_2 \frac{\text{size}}{f(P)} + \beta_1 + \cdots + \beta_n + c \\
& = a_1 \log(\text{size}) + a_2 \frac{1}{f(P) + \text{size}} + \beta_1 + \cdots + \beta_n + c \\
& = a_1 \log(\text{size}) + a_2 \frac{1}{1 + \text{size}/f(P)} + \beta_1 + \cdots + \beta_n + c.
\end{align*}
\]

It should be noted that although the second term in the last formula is not a linear function but a rational function of \( \frac{\text{size}}{f(P)} \), it can be approximated well by a linear function within some bounded domain. In fact, as shown in Fig. 2 the function \( y = 1/(1+x) \) decreases very slowly as \( x \) increases. It can be well approximated by a line if you limit the domain of \( x \) to an interval \((1/2, 2)\) for example.

The expression \( \frac{\text{size}}{f(P)} \) is obtained by approximating \( \frac{\text{test.defect}}{f(P)} \). Since both \text{test.defect} and \( f(P) \) represent the number of defects found in the program, their values should not be too different. Therefore the value of \( \frac{\text{size}}{f(P)} \) is expected to be within some bounded domain and the line approximation mentioned above is meaningful. When we actually apply our model, we will choose data whose defect density is limited within certain interval. We will validate the resulting models by actual project data in 4.1 and 4.2, where we limit \( \frac{\text{size}}{f(P)} \) to intervals \((1/2, 2)\) and \((1/4, 1)\) respectively.

From this argument we obtain the following formula.

\[
\begin{align*}
\log(\text{test.defect}) \\
& = a_1 \log(\text{size}) + a_2 \frac{\text{size}}{f(P)} + \beta_1 + \cdots + \beta_n + c.
\end{align*}
\]

It should be noted that the constant \( k \) appeared in the Iterative Prediction Model is included in the regression coefficient \( a_2 \) in this formula. It follows that we need to conduct regression analysis only once.

The above argument for the phase \( P \) of programming can be applied similarly to the phase \( DS \) of detailed design which is performed just before \( P \). More precisely an equation of the following form can be employed as the Quality Prediction Model:

\[
\begin{align*}
\log(\text{test.defect}) \\
& = a_1 \log(\text{size}) + a_2 \frac{\text{size}}{f(DS)} + \beta_1 + \cdots + \beta_n + c.
\end{align*}
\]

In this case the proposition (Construction of Iterative Prediction Model) in 3.1 will be modified as follows.

\textbf{(Construction of Iterative Prediction Model (for DS))}

Assume that the equation of the Effect Evaluation Model

\[
\text{test.defect} = \text{size}^{a_1} \exp(c + a_2 (DS) + \beta_1 + \cdots + \beta_n)
\]

holds without an error term, where the function \( d_{DS}(x) \) is defined as \( d_{DS}(x) = f(DS)/(f(DS) + x) \).

If \( t_{d_0} \) is a good enough approximation for \( \text{test.defect} \) whose error is within \( \varepsilon > 0 \), then

\[
\text{size}^{a_1} \exp(c + a_2 d_{DS}(t_{d_0}) + \beta_1 + \cdots + \beta_n)
\]

will give an approximation of \text{test.defect} such that the absolute value of its error is smaller than \(|a_2\varepsilon|/4\).

The number of defects at the programming phase \( f(P) \) is counted for the source code just like \text{test.defect}. Thus we can take an initial solution of \( f(P) + \text{test.defect} \) to be proportional to the development size \( \text{size} \) (KLOC).

The rest of the argument is the same as that for the programming phase \( P \). As a result, we obtain

\[
\begin{align*}
\log(\text{test.defect}) \\
& = a_1 \log(\text{size}) + a_2 \frac{\text{size}}{f(DS)} + \beta_1 + \cdots + \beta_n + c
\end{align*}
\]

as a formula for the Quality Prediction Model.

Furthermore we can apply this argument to one combined phase \( BS + FS + DS \). This phase is considered to be located right before the programming phase, too. In these three phases \( BS, FS, \) and \( DS \), peer reviews are conducted for documents. Hence, peer reviews in these phases are expected to have some common characteristics.

Let us summarize the resulting models for coding and document peer reviews.

\textbf{(Quality Prediction Models)}

The following equations hold.

1) In case of coding review,

\[
\begin{align*}
\log(\text{test.defect}) \\
& = a_1 \log(\text{size}) + a_2 \frac{\text{size}}{f(P)} + \beta_1 + \cdots + \beta_n + c,
\end{align*}
\]

![Fig. 2: Graph of \( y=1/(1+x) \) and a line approximation](image-url)

The above argument for the phase \( P \) of programming can be applied similarly to the phase \( DS \) of detailed design which is performed just before \( P \). More precisely an equation of the following form can be employed as the Quality Prediction Model:

\[
\begin{align*}
\log(\text{test.defect}) \\
& = a_1 \log(\text{size}) + a_2 \frac{\text{size}}{f(DS)} + \beta_1 + \cdots + \beta_n + c.
\end{align*}
\]

This case follows without an error term, where the function \( d_{DS}(x) \) is defined as \( d_{DS}(x) = f(DS)/(f(DS) + x) \).

If \( t_{d_0} \) is a good enough approximation for \( \text{test.defect} \) whose error is within \( \varepsilon > 0 \), then

\[
\text{size}^{a_1} \exp(c + a_2 d_{DS}(t_{d_0}) + \beta_1 + \cdots + \beta_n)
\]

will give an approximation of \text{test.defect} such that the absolute value of its error is smaller than \(|a_2\varepsilon|/4\).
2) In case of document review,
\[
\log(\text{test.defect}) = a'_1 \log(\text{size}) + a'_2 \text{size}/(f(BS)+f(FS)+f(DS)) + \\
\beta'_1 + \cdots + \beta'_n + c',
\]
where \(a_1, a_2, c, a'_1, a'_2, c'\) are constants, \(\beta_1, \cdots, \beta_n, \beta'_1, \cdots, \beta'_n\) are terms representing factors other than peer review performance.

It should be noted that the right hand side of the Quality Prediction Models are actually computable at the \(P\) or \(DS\) phases. In fact at these phases the development size \(\text{size}\) is either already determined or almost determined and precisely predictable.

Furthermore, in the target organization baselines for peer review performance (standards value and permissible ranges) are established and peer review processes are controlled with them[7]. The measures for these baselines include review rate and defect density. The Quality Prediction Models help a project to realize how many defects should be detected by peer reviews in order to achieve the project’s quality objective (given in terms of defect density at the testing phase). The baselines and peer review process control with them are used to ensure enough defects are detected by peer reviews.

4. Validation by Actual Project Data

The Quality Prediction Model proposed in the previous section is derived under the hypothesis that the equation of the Effect Evaluation Model holds without the error term. In reality this hypothesis is not true. However, we can expect the Quality Prediction Model is valid as a regression equation. In order to show this, we conduct regression analysis on actual project data. We use the same 35 project data as used for the validation of the Effect Evaluation Model. The results of the regression analysis are given below.

4.1 Case of Coding Review

In the target organization projects define their quality objectives through ‘bag rate’ which is the defect density at the testing phase \(T\). We select the condition to limit the line approximation described in 3.2 to be that the defect density at the programming phase \(P\) should be between \(1/2\) and 2 times of the project’s quality objective.

The rationale for this choice is that \(\text{test.defect}/f(P)\) should take values around 1 as explained in 3.2. Actual width of the interval is determined so that we can keep enough precision while we can employ as many data as possible. Data from 23 projects satisfy this condition out of 35 projects.

As a result, we obtain the following regression equation and results.

\[
\text{test.defect} = 6.08 \cdot \text{size}^{1.003} \exp(-0.0403 \log(\text{base.size}+1)) + 0.616 \cdot \text{size}/f(P) - 0.597\cdot PM - 0.496SF,
\]

null deviance: 948.95, residual deviance: 52.196, degree of freedom: 17, pseudo \(R^2\): 94.5%. Here, \(\text{base.size}\) represents the size of the base system (KLOC). We add 1 to this value and take the logarithm as we did for the Effect Evaluation Model. The explanatory variables \(PM\) and \(SF\) are dummy variables concerning risks about project leader’s experience and freedom of system development respectively. As in the Effect Evaluation Model we use quasi-Poisson distribution instead of Poisson distribution, because we observed over dispersion.

In this equation \(\text{size}/f(P)\) has a positive regression coefficient. It follows that if you increase \(f(P)\) while keeping the value of \(\text{size}\) then \(\text{test.defect}\) will decrease, which means the product quality is enhanced. This property is practically important, because we can take remedy actions to increase the number of detected defects, if the predicted quality is not satisfactory. Furthermore it is possible to evaluate these activities quantitatively.

The term containing \(\text{base.size}\) has negative coefficient, which suggests that if you have some experience of developing the same or related system, the quality will be enhanced. Both of the two dummy variables \(PM\) and \(SF\) have negative coefficients, which suggests that if the risk evaluation results are higher, then the quality will be enhanced. This result seems to be counterintuitive. However, as we explained in section 2, in the target organization projects are required to take some specific action, if the risk evaluation result is 4 or higher. There is a possibility that the quality is enhanced as a result of this action.

Fig. 3 is the scatter diagram of the prediction of the number of defects detected at the testing phase and the actuals. Most points are located near the diagonal line and the variation is not too large.

4.2 Case of Document Review

We construct a Quality Prediction Model for the combined phase \(BS+FS+DS\). We select the condition to limit the line approximation to be as follows: The number of defects
detected by peer reviews on documents should be between 1 and 4 times of the number of defects expected to be found at the testing phase. It should be noted that the latter number is calculated by the project’s quality objective and estimated development size.

Since the numbers of defects over three phases are summed up, both the upper and lower limits of the condition is greater than those of coding review. These limits are determined so that we can keep enough precision while we can employ as many data as possible just like those of coding review. In terms of the interval to limit the domain of the function \( y = 1/(1 + x) \), this means we take an interval \((1/4, 1)\). It should be noted that we take the inverse value, because \( x = test\_defect / (f(BS) + f(FS) + f(DS)) \). Data from 22 projects satisfy this condition out of 35 projects.

As a result, we obtain the following regression equation and results.

\[
\begin{align*}
\text{test}\_\text{defect} &= 6.73size^{1.09} \exp(1.81size / (f(BS) + f(FS) + f(DS)) - 0.81RC - 0.39RP), \\
\text{null deviance: } &4030.46, \text{ residual deviance: } 184.66, \text{ degree of freedom: } 17, \text{ pseudo } R^2 : 95.4\%. 
\end{align*}
\]

The explanatory variables \( RC \) and \( RP \) are dummy variables concerning whether the requirements are clearly documented and whether the current system development replaces the existing system which is developed by other companies.

We use quasi-Poisson distribution instead of Poisson distribution, because we observed over dispersion. The regression coefficient suggests that the quality will be enhanced, if we increase the number of defects detected by peer reviews in this case, too. Similarly, there is a tendency that the larger the risk evaluation result is, the higher the quality will be.

Fig. 4 is the scatter diagram of the prediction of the number of defects detected at the testing phase and the actuals. Judging from this graph, the data point at the upper right might affect strongly on the regression result. We investigated the Cook’s distance. However, every data has 0.5 or lower distance. We can not find any evidence that the data affects strongly.

Fig. 5 shows an enlarged graph of the projects which detected 300 or less defects. Comparing this graph with Fig.3 which is drawn on the same scale, you can see this graph has larger variation. This may be because the programming phase \( P \) is still left when a project just completes the \( BS + FS + DS \) phase.

In fact, we found the following fact by investigating the data. There are two circled groups of projects in Fig.5. The projects in the group located below the diagonal line detected relatively more defects in coding reviews. On the other hand, the projects in the group located above the diagonal line detected considerably fewer defects in coding review. This observation would imply that if you have a bad result by the Quality Prediction Model for document reviews, you still have an opportunity to improve the quality by conducting intensive peer reviews in the coming \( P \) phase.

Since we have overdispersions for both coding and document reviews, there is a possibility that the Poisson distribution is not appropriate. We conduct the conformance testing for the regression residuals. As a result, we get the following p-values: 0.26 for coding reviews and 0.38 for document reviews. Therefore, the null hypothesis that the distribution is Poisson is not rejected for both cases.

### 4.3 Discussion on Precision

We evaluate the precisions of the predictions by cross validation method. That is, select one subset of project data from the data set being analyzed, construct a model with the data set excluding the selected data, apply the model to the selected data to get prediction, and measure the difference between prediction and actual for the selected data. We apply this procedure to the 23 data of coding reviews and the 22 data of document reviews. As a result, we get the following average relative error: 18.6% for coding reviews and 24.0% for document reviews.
Since we can not find any publicized data for the precision of estimates of the number of defects at testing phase, it is hard to evaluate our results objectively. However, by our experience, it is not unusual to have 10–20% relative errors for an expert project manager to estimate the defect density at testing phase, even if he or she has experience of developing similar systems.

If one uses the Quality Prediction Model in this paper, one can get predictions with similar or a little bit worse precision even if one is not an expert or unfamiliar with the kind of system being developed. Thus, we consider our result is good enough to apply real-world development for the target organization and perhaps a similar quality prediction model could be derived, but with other weights and possibly a different set of input variables for other organizations.

5. Related Researches

There are several researches to predict quality from the project performance. Gaffney[11] and Kan [12] proposed to use Rayleigh curve to estimate the latent defects in the field. This method is validated by real world project data. However, the data they used are from very large size projects following rigid waterfall lifecycle phases which are not very popular these days. What they predict is not the number of defects at the testing phase but the number of latent defects in the field.

COCOMO II[4] is a family of various estimation models for software development including quality prediction. As for quality prediction, it employs log-transformed linear regression with several explanatory variables including development size. The explanatory variables other than development size use expert’s opinion obtained by delphi method. Unlike the number of defects used in this paper, these variables do not use measured performance data.

Nakajima and Azuma[13] proposes a model to show how the quality cost will affect on overall development cost, based on ‘Management model of total number of the defects.’ A part of the formulation of their model is similar to ours at least formally. It appears that the estimating functions they propose are not validated through actual data. What they propose is a simulation model whose input are objective values of a project. A project compares the output of the model with actual project data and take corrective action if necessary. It does not predict the future outcome from the current project data.

Morgan and Knafl [14] proposes a regression model to predict residual fault density after the software system is released. Unlike the result of this paper, they do not predict the number of defects at the testing phase from the performance data at the upper stream phases.

6. Conclusion

We proposed a method to construct a model to predict the number of defects at the testing phase from the peer review performance data.

Firstly, we established a model to relate the number of defects at the testing phase with the defect removal rate of peer reviews at each development phase through regression analysis using Generalized Linear model. Secondly, we transform the model into another so that all the input data are available at an earlier development stage, i.e., peer review performance data and risk analysis results. Finally, we validated the proposed model with real world project data and found that the model can predict with $\pm 18\%$ to $\pm 24\%$ relative errors, which we consider good enough for practical application.

This model has the following advantages: (1) It can predict the quality at early point in time. (2) It predicts the number of defects at the testing phase. And hence, we can know not only the quality improvement effect but also the cost reduction effect (through the reduction of rework). (3) Projects can take concrete corrective actions, if the prediction result is not satisfactory.

6.1 Acknowledgements

We would like to express our hearty thanks to Dr. Micheael Konrad of the Software Engineering Institute for his valuable comments.

References