# On Estimating Testing Effort Needed to Assure Field Quality in Software Development

Osamu Mizuno[†], Eijiro Shigematsu[†], Yasunari Takagi[†‡], and Tohru Kikuno[†]
[†]Graduated School of Information Science and Technology, Osaka University, Japan.
E-mail: {o-mizuno,kikuno}@ist.osaka-u.ac.jp
[‡]OMRON Corporation, Japan.
E-mail: taka@biwa.kusatsu.omron.co.jp

## Abstract

In the practical software development, software quality is generally evaluated by the number of residual defects. To keep the number of residual defects within a permissible value, too much effort is often assigned to software testing.

In this paper, we try to develop a statistical model to determine the amount of testing effort which is needed to assure the field quality. The model explicitly includes design, review, and test (including debug) activities.

Firstly, we construct a linear multiple regression model that can clarify the relationship among the number of residual defects and the efforts assigned to design, review, and test activities. We then confirm the applicability of the model by statistical analysis using actual project data.

Next, we obtain an equation based on the model to determine the test effort. As parameters in the equation, the permissible number of residual defects, the design effort, and the review effort are included. Then, the equation determines the test effort that is needed to assure the permissible residual defects. Finally, we conduct an experimental evaluation using actual project data and show the usefulness of the equation.

## 1 Introduction

The importance of the software quality has been increasing for the last decade. In order to measure the software quality, many metrics and methodologies have been proposed up to now[5, 7, 14]. Among them, the number of residual defects is frequently used since it is easily understandable and deeply concerned with the needs in the software development organization [7].

On the other hand, various approaches have been proposed to decrease the number of residual defects [4, 7]. The most direct approach is to perform enough testing. However, improving the overall development process is more desirable to achieve high software quality [2, 6, 9]. For instance, constructing rigid specification, introducing review activity, and determining feasible development plans are such improvement activities.

In the actual industry, the software testing effort has been usually determined by using the past experience. In order to improve such a situation, many studies have been proposed to estimate appropriate testing effort. For instance, the software reliability growth model(SRGM) is one of the best-known approach to determine optimal release point[8]. However, since the SRGM uses the whole activity data from the beginning to the end of defect detection, it cannot be used before the testing begins. On the other hand, Takahashi et al. showed some metrics and a linear correlation model to predict the number of errors[12]. Though they estimated the number of errors successfully, their metrics are mainly product metrics. Moreover, their aim was not to estimate the testing effort, but predict the number of residual faults in software.

We present an estimating method of the test effort to assure field quality. Here, assuring the field quality means to keep the number of residual defects below the permissible number determined a priori.

We focus on a timing when the development goes to the end of its coding activity. We thus assume that the efforts needed for design and review activities have already been known, since these activities were over. Under such a condition, we construct a model to estimate the testing effort considering the field quality. The model makes it possible to estimate the testing effort dynamically before the test activities begin. Additionally, the testing effort estimated by the model is the one necessary to assure field quality. As a result, we can expect even the improvement of productivity for testing activities. Furthermore, since the model itself is quite simple, the developers can understand the model easily.

In order to establish the proposed method, we firstly develop a linear multiple regression model that explains the number of residual defects by a linear function of design, review, and test efforts. To do so, as a preliminary investigation, we statistically confirm that software development efforts such as design, review, and test efforts have much effect on the number of residual defects by using actual project data.

Next, we obtain an equation, based on the linear multiple regression model, to determine the test effort. Here, we assume that the number of permissible residual defects is given, and the design and the review efforts are specified. We then conduct an experimental evaluation using actual project data and showed the usefulness of the equation.

The rest of this paper is organized as follows. Section 2 describes the preliminaries of this study, such as process model and software metrics. In Section 3, we explain an approach, which consists of three phases. Then, in Section 4, we investigate the relationship between the effort assigned to each activity and the number of residual defects. Next in Section 5, we construct a model that can clarify the relationship among the number of residual defects and the efforts assignment. In Section 6, we obtain an equation to determine the test efforts that is needed to assure the permissible residual defects and conduct an experimental evaluation using actual project data. Finally, Section 7 summarizes this paper.

## 2 Preliminaries

### 2.1 Process model

The projects targeted in this paper are the development of computer control systems with embedded software in a certain company. The systems are related to retail systems. Such embedded software implements rather complex functions dealing with many sensors, actuators, and control signals including various kinds of interrupts. Furthermore, since it is delivered in the form of LSI chips, modification of defects after delivery is very expensive. Thus, high quality is especially required for the embedded software.

We use the actual project data of 38 projects, which have already finished their development. The characteristics of these projects are as follows:

1. The projects started their development in 1996 and 1997.

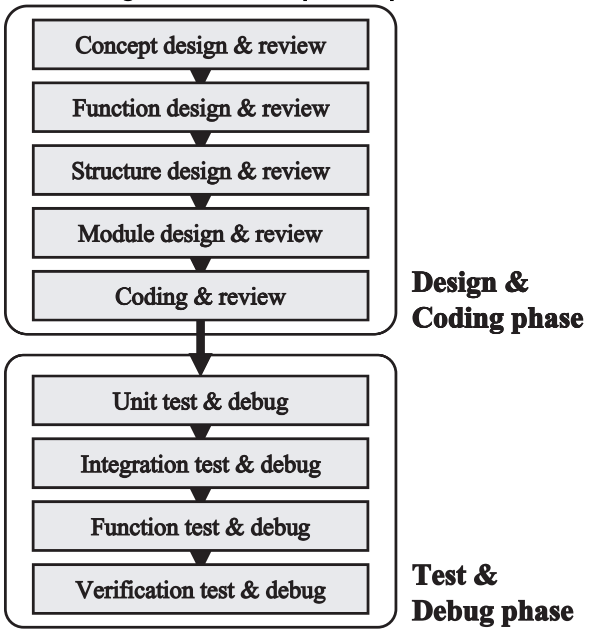
2. The development efforts of these 38 projects range from 7 to 35 person-months. The average effort is about 14 person-months.

3. At least 15% of the total efforts for design and coding activities was assigned to review activities.

These projects can be considered very similar ones, since their objective is to produce almost the same software systems to different customers and thus their development process is almost the same.

In the target projects, many kinds of computer systems with embedded software are developed mainly using C language. The products are developed under a development process as shown in Figure 1. The development process is an ordinal waterfall model.

Figure 1. Development process



The development process consists of two successive phases, design & coding phase and test & debug phase. The design & coding phase is divided into five stages: Concept design, Function design, Structure design, Module design, and Coding. The test & debug phase is divided into four stages: Unit test & debug, Integration test & debug, Function test & debug, and Verification test & debug.

One characteristic of the design & coding phase is that review activity is introduced after each design activity. Review activity is an approach that enables the detection and correction of defects in software artifacts as soon as these artifacts are created. The review activity not only improves the quality of the artifacts but also helps software development organizations reduce their cost of producing software [1, 3]. In the review activity, the documents should be distributed to the persons concerned in the company, and then review results should be returned to developers via managers (this review activity is called peer review [1]). They established several guidelines for the review activity. One of them suggests that at least 15% of the total efforts for design & coding phase should be assigned to review activities [11].

The test & debug phase consists of the repetition of a pair of test activity and debug activity. Test activity is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software items. Debug activity is the process to detect, locate, and correct faults [10]. Developers are directed to record all defects that are detected by the test activity and removed by the debug activity [13]. In order to improve and assure the quality of the software product, the product needs to be tested sufficiently.

## 2.2 Software metrics

In this subsection, we define some software metrics[5] used in this paper. Five software process metrics related to effort of development phases are defined as follows:

(1) $E_{design}$ (design effort) : the effort spent on the design and coding activities in design & coding phase (person-day).

We define the design effort as the sum of the effort spent on the design and coding activities.

(2) $E_{review}$ (review effort) : the effort spent on the review activities in design & coding phase (person-day).

We define the review effort as the sum of the effort spent on each review activity.

(3) $E_{test}$ (test effort) : the effort spent on the test and debug activities in test & debug phase (person-day).

We define the test effort as the sum of the effort spent on test activity and the effort spent on debug activity.

(4) $R_{design/total}$ (ratio of the design effort to the total development effort):

$$R_{design/total} = \frac{E_{design}}{E_{design} + E_{review} + E_{test}}$$

This metric $R_{design/total}$ stands for the ratio of the design effort to the sum of the efforts assigned to the design activity, the review activity, and the test & debug activity.

(5) $R_{test/total}$ (ratio of the test effort to the total development effort):

$$R_{test/total} = \frac{E_{test}}{E_{design} + E_{review} + E_{test}}$$

This metric $R_{test/total}$ stands for the ratio of the test effort to the sum of the efforts assigned to the design activity, the review activity, and the test & debug activity.

A metric related to the software product is defined as follows:

(6) $D_{field}$ (number of field defects) : the number of field defects after six months after delivery. Here, a field defect means a failure found in the customer's site.

We define $D_{field}$ as a measure of software quality. Although the number of defects is not a sufficient metric for software quality, $D_{field}$ has been considered to be a quality metric in this company.

## 2.3 Actual project data

In each project, effort data and defect data are recorded manually and stored in workstations by each staff member. Then, they are collected and summed up by the project leader, and validated by the manager. Field defect data are reported by a quality assurance staff member, translated into a defect-based number by the project leader, and also validated by the manager. All the validated data are sent to the software engineering process group (SEPG), who analyzes it and reports back to the project team and development organization [11].

Tables 1 and 2 show the actual effort data in 1996 and 1997, respectively. Table 3 summarizes the average and the standard deviation of $E_{design}$, $E_{review}$, and $E_{test}$. Note that these projects were selected according to some criteria (such as the product size, cost, and so on) defined in the company.

**Table 1. Actual efforts of the projects in 1996**

| No. | $E_{design}$ | $E_{review}$ | $E_{test}$ |
|---|---|---|---|
| Y96-1 | 81.0 | 22.9 | 73.1 |
| Y96-2 | 122.4 | 28.6 | 190.0 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| Y96-19 | 428.0 | 86.0 | 130.0 |

**Table 2. Actual efforts of the projects in 1997**

| No. | $E_{design}$ | $E_{review}$ | $E_{test}$ |
|---|---|---|---|
| Y97-1 | 138.5 | 32.5 | 77.9 |
| Y97-2 | 81.2 | 22.5 | 128.3 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| Y97-19 | 150.0 | 28.0 | 43.0 |

## 3 An Approach to Estimation

In this paper, we suggest a timing when the design & coding phase in Figure 1 is finished. We thus assume that

**Table 3. Fundamental statistics for $E_{design}$, $E_{review}$, and $E_{test}$**

| Variable | Average | Std. Dev. |
|---|---|---|
| $E_{design}$ | 120 | 77.0 |
| $E_{review}$ | 28 | 17.5 |
| $E_{test}$ | 108 | 58.7 |

the efforts needed for design and review activities have already been known. Under such a condition, we construct a model to estimate the testing effort needed to assure the permissible number of field defects. Straightforwardly speaking, we estimate $E_{test}$ from $E_{design}$ and $E_{review}$ according to the field quality.

In order to establish the method, we perform the following three phases:

**Phase 1:** (Investigating the efforts' effect on the number of field defects.)

We clarify the relationship between the effort assigned to each activity and the number of field defects. Additionally, we prove the correctness of the correlation by the statistical tests.

**Phase 2:** (Construction of a regression model for the number of field defects.)

According to the result obtained in Phase 1, we construct a multiple regression model. This model clarifies the multivariate relationship between efforts for activities and the number of field defects. Then, we prove the correctness of the model by the statistical test.

**Phase 3:** (Estimation of testing effort to assure field quality.)

By transforming the regression model, we construct an equation to calculate the testing efforts when the design and review efforts, and the permissible number of field defects are given.

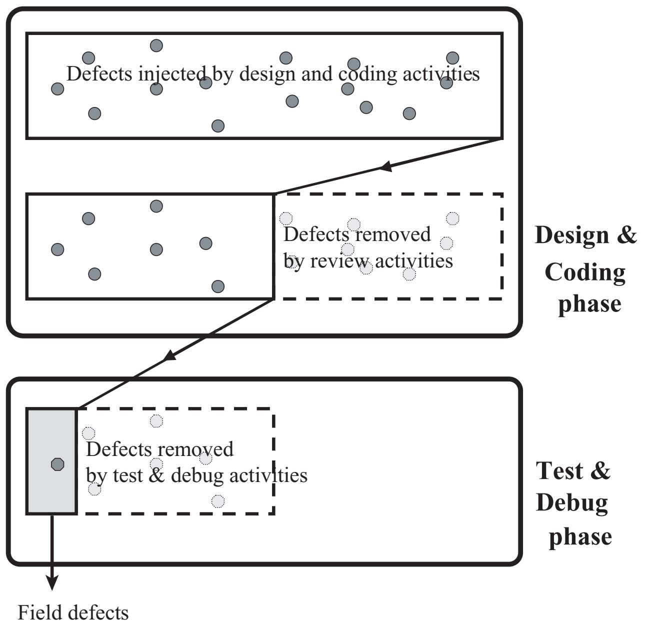
For the verification of Phase 1 and 2, we use the data of projects performed in 1996 and 1997 (that is, all projects shown in Tables 1 and 2). But in Phase 3, we apply the data of projects in 1996 shown in Table 1 for model construction, and for an experimental application of the model, we apply the data of 1997 shown in Table 2.

## 4 Investigating Effects of Efforts (Phase 1)

### 4.1 Scheme of the proposed method

Figure 2 shows a scheme for defect injection and defect removal. In this paper, we assume that defects are injected during the design and coding activities. On the contrary, those defects are detected and removed by review, test, and debug activities. The remaining defects are delivered to customers as field defects.

**Figure 2. Defects injected and removed process**


Defects injected by design and coding activities
Defects removed by review activities
Design & Coding phase
Defects removed by test & debug activities
Test & Debug phase
Field defects

In this scheme, we can use three kinds of effort data $E_{design}$, $E_{review}$, and $E_{test}$ and a field defect data $D_{field}$. In the next subsection, we analyze their relationship.

### 4.2 Correlation between metrics

Firstly, we analyze the effect of software development efforts such as design, review, and test efforts on the number of field defects. We make the following three hypotheses with respect to the relationship between the number of field defects and effort assignment.

$H_1$ : An increase in the effort assigned to the design and the coding activities increases the number of field defects.

$H_2$ : An increase in the effort assigned to the review activity decreases the number of field defects.

$H_3$ : An increase in the effort assigned to the test and the debug activities decreases the number of field defects.

#### 4.2.1 $D_{field}$ and $R_{design/total}$

We investigate the correlation coefficient between $E_{design}$ and $D_{field}$ to confirm the hypothesis $H_1$. Since the date of delivery is promised and the developers have to finish the project within a given duration in the target company, the increase in the effort assigned to the design activity can decrease the efforts assigned to the review activity and test & debug activity. Therefore, we analyzed the relationship between the ratio of effort assigned to the design activity and the number of field defects.

The value of Spearman's rank correlation coefficient between $R_{design/total}$ and $D_{field}$ is 0.535 ($p < 0.01$). It implies that the correlation between $R_{design/total}$ and $D_{field}$ is rather strong. We thus confirm the hypothesis $H_1$.

#### 4.2.2 $D_{field}$ and $R_{review/design}$

The review process improvement activity was established in the company and the software quality changes caused by the review improvement were evaluated [9, 11]. For the target projects in this paper, we consider the relationship between the ratio of effort assigned to the review activity and the number of field defects in order to confirm the hypothesis $H_2$.

We analyzed the value of Spearman's rank correlation coefficient between $R_{review/design}$ and $D_{field}$. The coefficient is $-0.385$ ($p < 0.05$). It implies that there is a certain degree of correlation between $R_{review/design}$ and $D_{field}$, since the value of correlation coefficient is not so large.

#### 4.2.3 $D_{field}$ and $R_{test/total}$

We consider the relationship between the ratio of effort assigned to the test activity and the number of field defects in order to confirm the hypothesis $H_3$.

The value of correlation coefficient between $R_{test/total}$ and $D_{field}$ is $-0.491$ ($p < 0.01$). It implies that the negative correlation between $R_{test/total}$ and $D_{field}$ is rather strong.

This result shows that the increase in the effort assigned to the test activity is still related to the decrease in the number of field defects. Therefore, it is valuable to analyze the relationship between efforts assigned and the number of field defects, and determine the test effort needed to assure permissible software quality.

## 5 Construction of Model (Phase 2)

### 5.1 Regression model for $D_{field}$

In order to determine the amount of testing effort which is needed to assure the field quality, we construct a linear multiple regression model, described by Equation (1). Equation (1) can clarify the relationship, which is derived in subsection 4.2, among the number of field defects and the efforts assigned to design, review, and test activities.

$$D_{field} = b_0 E_{design} - b_1 E_{review} - b_2 E_{test} \qquad (1)$$
$$(b_0 > 0, b_1 > 0, b_2 > 0)$$

Since the design activities can be viewed as the process in which defects are injected, the regression coefficient of $E_{design}$ is plus. On the contrary, since the review activities and the test & debug activities can be viewed as the process in which defects are removed, both regression coefficients of $E_{review}$ and $E_{test}$ are minus.

### 5.2 Estimation of coefficients

By applying the ordinary least squares method to the 38 project data in Tables 1 and 2, we estimated the parameters in Equation (1). The result of estimation is summarized in Table 4.

**Table 4. Estimated values of parameters**

| Independent variable | Estimated coefficient | Standard coefficient | p-value |
|---|---|---|---|
| $E_{design}$ | 0.045 | 1.29 | < 0.0001 |
| $E_{review}$ | 0.091 | 0.59 | 0.048 |
| $E_{test}$ | 0.012 | 0.27 | 0.004 |

The estimated coefficients are as follows:

$$b_0 = 0.045, b_1 = 0.091, b_2 = 0.012$$

It is shown that signs of $b_0$, $b_1$, and $b_2$ are plus. We thus confirm that each of these three coefficients is significant for Equation (1).

Coefficient of determination ($R^2$) is a measure of how well the model explains the variance in the dependent variable, and it takes a value from 0 to 1. Statistical analysis revealed $R^2$ of 0.722, which implies that the model explains 72.2 percent of the variation observed in $D_{field}$. When $R^2$ is more than 0.4, we can generally consider the value as high coefficient. Since the value of $R^2$ is very high, we confirm that the constructed Equation (1) explains the variance of the number of field defects very well.

## 6 Estimation of Test Effort (Phase 3)

### 6.1 Proposed method

From Equation (1), we obtain the following Equation (2):

$$E_{test} = (D_{field} - b_0 E_{design} + b_1 E_{review})/(-b_2) \qquad (2)$$

Equation (2) expresses the test effort ($E_{test}$) as a function of the number of field defects ($D_{field}$), the design effort ($E_{design}$), and the review effort ($E_{review}$).

We propose a method for determining the test effort needed to assure permissible software quality by using Equation (2) and past project data. It consists of the following four steps:

**Step 1:** Calculation of $b_0$, $b_1$, and $b_2$ in (2).

In this step, past project data are prepared as training data for the regression model. Then, $b_0$, $b_1$, and $b_2$ are estimated using the ordinary least squares method. The coefficients are tested statistically.

**Step 2:** Assignment of a given constant to $D_{field}$ in (2).

In this step, we determine $D_{field}$ based on the requirements for the product or the development plan so that the quality of the product can be assured. We call the determined $D_{field}$ as the permissible number of field defects and denote it by $\hat{D}_{field}$.

**Step 3:** Assignment of actual values to $E_{desgin}$ and $E_{review}$ in (2).

In this step, the design effort and the review effort of the target project are set as $E_{design}$ and $E_{review}$ respectively. According to the progress of a project, $E_{design}$ and $E_{review}$ are changed in the following way, if necessary.

- If the design & coding phase has been completed, then the actual data of the design effort and the review effort are $E_{design}$ and $E_{review}$.
- If the design & coding phase has not been completed or the actual data are not measured, then estimated values are assigned to $E_{design}$ and $E_{review}$.

**Step 4:** Calculation of $E_{test}$ by (2).

We calculate the value of $E_{test}$ by Equation (2). We represent this calculated value by $\hat{E}_{test}$ and call the test effort needed to assure the permissible software quality. Then, $\hat{E}_{test}$ is defined by the following Equation (3).

$$\hat{E}_{test} = (\hat{D}_{field} - b_0 E_{design} + b_1 E_{review})/(-b_2) \qquad (3)$$

As mentioned before, in this paper, assuring the field quality means to keep the number of field defects below the permissible number determined a priori. Therefore, if the test effort of the project is more than $\hat{E}_{test}$, then the number of field defects is expected to be less than $\hat{D}_{field}$.

## 6.2 Experimental evaluation

In Step 1, we calculated the parameters $b_0$, $b_1$, and $b_2$ in model (1) by using project data in 1996. The values of $E_{test}$, $E_{design}$, and $E_{review}$ in 1996 are shown in Table 1. Here, we cannot publish the actual values of $D_{field}$, since these values are confidential.

The coefficients calculated were as follows:

$$b_0 = 0.039, b_1 = 0.069, b_2 = 0.011$$

Since the coefficient of determination ($R^2$) is 0.743, we can say that the estimated coefficients explain the small variance in the number of field defects very well.

In Step 2, we determined the permissible number of field defects. As the company intends to make lower the number of field defects as much as possible, we determined the test effort to keep the number of field defects less than 1. Therefore, we set the permissible number of field defects $\hat{D}_{field} = 1$.

As a result, we constructed Equation (4) from Equation (3).

$$\hat{E}_{test} = (1 - 0.039 E_{design} + 0.069 E_{review})/(-0.011) \qquad (4)$$

In Step 3, we determined to assign the actual values of the design and the review efforts of each project in 1997 (shown in Table 2) to $E_{design}$ and $E_{review}$ in Equation (4), respectively.

Then in Step 4, we were able to calculate $\hat{E}_{test}$ for the 19 projects in 1997 from Equation (4). The calculated values of $\hat{E}_{test}$ are shown in Table 5.

In this table, $\hat{E}_{test}$ represents the test effort calculated by proposed method. "Comparison" shows the comparison result of actual $E_{test}$ (please note that actual $E_{test}$ is given in Table 2) and $\hat{E}_{test}$. In this column, "○" indicates that $E_{test}$ is more than $\hat{E}_{test}$, and "×" indicates that $E_{test}$ is less than $\hat{E}_{test}$. Again, please note that projects in Tables 1 and 2 were obtained according to some criteria defined in the company.

By summarizing Table 5, we classified the projects according to the following two conditions:

$C_1$ **:** $E_{test}$ is more than $\hat{E}_{test}$.

$C_2$ **:** No field defect was found.

The projects were classified as shown in Table 6. Table 6 provides the following two observations:

- There exist eight projects in which the test effort is more than calculated $\hat{E}_{test}$. For seven projects out of the eight projects, no field defect is detected.
- There exists eleven projects in which the test effort is less than calculated $\hat{E}_{test}$. For eight projects out of the eleven projects, some field defects are detected.

Table 5. Comparison of $\hat{E}_{test}$

| No. | $\hat{E}_{test}$ | Comparison |
|---|---|---|
| Y97-1 | 196.3 | × |
| Y97-2 | 55.8 | ○ |
| Y97-3 | 73.1 | × |
| Y97-4 | 85.6 | × |
| Y97-5 | 29.5 | ○ |
| Y97-6 | 4.0 | ○ |
| Y97-7 | 43.5 | ○ |
| Y97-8 | 40.2 | ○ |
| Y97-9 | 79.7 | ○ |
| Y97-10 | 59.2 | ○ |
| Y97-11 | 198.4 | × |
| Y97-12 | 42.5 | ○ |
| Y97-13 | 190.0 | × |
| Y97-14 | 251.3 | × |
| Y97-15 | 105.7 | × |
| Y97-16 | 124.6 | × |
| Y97-17 | 145.3 | × |
| Y97-18 | 440.1 | × |
| Y97-19 | 265.3 | × |

Table 6. Classification of experimental results

| | | Field defects $D_{field}$ | | Total |
|---|---|---|---|---|
| | | not found | found | |
| Testing effort $E_{test}$ | More than $\hat{E}_{test}$ | 7 | 1 | 8 |
| | Less than $\hat{E}_{test}$ | 3 | 8 | 11 |
| Total | | 10 | 9 | 19 |

In order to confirm the statistical significance of these observations, we analyzed the relationship between $C_1$ and $C_2$ by using Fisher's exact probability test. Fisher's exact probability test calculates an exact probability value for the relationship between two dichotomous variables, as found in a two by two cross table like Table 6. For Table 6, the Fisher's test would determine whether the two groups, $E_{test}$ is more than $\hat{E}_{test}$ or less than $\hat{E}_{test}$, differ significantly in the proportion of "not found" and "found". The null hypothesis is that $C_1$ and $C_2$ are independent. The level of significance is chosen as 0.05. The calculated p-value of Fisher's exact test for the table becomes 0.0198. Then, the null hypothesis is rejected at the 0.05 level.

As a result, we can say that there are some correlation between the estimated test efforts and field defects. That is, the calculated testing efforts $\hat{E}_{test}$ can be a threshold of the testing efforts to keep the number of field defects less than $\hat{D}_{field} = 1$.

## 7 Conclusion

We proposed a method to determine efforts that should be assigned to the test & debug phase to assure the permissible software quality. To do so, firstly, we constructed a linear multiple regression model that explains the relationship between the number of field defects and the development efforts. Several statistical analyses using the actual project data confirmed that the constructed model nicely explains the data of target projects. Then, in order to calculate the test effort needed to assure permissible quality, we got an equation by transforming the constructed regression model.

As a result of experimental evaluation using actual project data, we confirmed that the following two results: (1) As for projects having larger test effort than calculated, no field defect was detected, and (2) As for projects having test effort less than calculated effort, some field defects remained.

However, this result only shows that our proposed method is applicable to the targeted projects in a certain company. To generalize it, we need to apply the proposed method to more data of other software projects. It is one of our most important future works.

## References

[1] D. B. Bisant and J. R. Lyle. A two-person inspection method to improve programming productivity. *IEEE Trans. on Software Engineering*, 15(10):1294–1304, 1989.

[2] M. Diaz and J. Sligo. How software process improvement helped motorola. *IEEE Software*, 14(5):75–81, 1997.

[3] M. E. Fagan. Advances in software inspections. *IEEE Trans. on Software Engineering*, 12(7):744–751, 1986.

[4] N. E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Trans. on Software Engineering*, 25(5):675–689, 1999.

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

[6] W. S. Humphrey. *A Discipline for Software Engineering*. Addison-Wesley, MA, 1995.

[7] M. S. Krishnan and M. I. Kellner. Measuring process consistency: Implications for reducing software defects. *IEEE Trans. on Software Engineering*, 25(6):800–815, 1999.

[8] J. D. Musa, A. Iannino, and K. Okumoto. *Software reliability: measurement, prediction, application*. McGraw-Hill, 1987.

[9] K. Sakamoto. *A study of software process improvement and quality control based on analyses of actual project data*. PhD thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, 2000. NAIST-IS-DT9861009.

[10] Standards Coordinating Committee of the IEEE Computer Society. *IEEE Std 610.12-1990*. The Institute of Electrical and Electronics Engineers, Inc., 1990.

[11] Y. Takagi, T. Tanaka, N. Niihara, K. Sakamoto, S. Kusumoto, and T. Kikuno. Analysis of review's effectiveness based on software metrics. In *Proc. of 5th International Symposium on Software Reliability Engineering*, pages 34–39, 1995.

[12] M. Takahashi and Y. Kamayachi. An empirical study of a model for program error prediction. In *Proc. of 8th International Conference on Software Engineering*, pages 330–336, 1985.

[13] T. Tanaka, K. Sakamoto, S. Kusumoto, K. Matsumoto, and T. Kikuno. Improvement of software process by process description and benefit estimation. In *Proc. of 17th International Conference on Software Engineering*, pages 123–132, 1995.

[14] S. Yamada and M. Takahashi. *Introduction to software management model*. Kyoritsu books (in Japanese), 1993.