# Statistical Analysis of Time Series Data on the Number of Faults Detected by Software Testing

Sousuke AMASAKI, Takashi YOSHITOMI, Osamu MIZUNO, Tohru KIKUNO, and Yasunari TAKAGI
Graduate School of Information Science and Technology, Osaka University, Japan
E-mail: amasaki@ics.es.osaka-u.ac.jp, {o-mizuno,kikuno,y-yakagi}@ist.osaka-u.ac.jp

## Abstract

*According to a progress of the software process improvement, the time series data on the number of faults detected by the software testing are collected extensively. In this paper, we perform statistical analyzes of relationships between the time series data and the field quality of software products.*

*At first, we apply the rank correlation coefficient $\tau$ to the time series data collected from actual software testing in a certain company, and classify these data into four types of trends: strict increasing, almost increasing, almost decreasing, and strict decreasing. We then investigate, for each type of trend, the field quality of software products developed by the corresponding software projects. As a result of statistical analyses, we showed that software projects having trend of almost or strict decreasing in the number of faults detected by the software testing could produce the software products with high quality.*

**Keywords:** *software testing, software quality, statistical analysis.*

## 1 Introduction

Reducing the cost for software maintenance is recognized as a serious problem, since the lifetime of software becomes long and software is updated many times according to new requirements. It is well known that one of the fundamental causes for serious excess of cost is the remaining faults in the software. Thus, reducing the number of the remaining faults at the development is very important [1,7].

A straightforward approach is to perform sufficient testing during the development. In order to manage the testing effectively, a software reliability growth model (SRGM) is proposed and really applied to many actual projects [10]. Since the SRGM needs the data on the number of faults detected by the testing activities, software metrics are defined and these quality data are extensively collected.

In the typical software development projects, the software testing consists of the unit test, integration test, function test, and validation test. Additionally, these tests are examined successively. Therefore, we can get time series data from these successive testing activities. However, as far as we know, these data have been analyzed individually (that is, data for each unit test, integration test, function test, or validation test were analyzed), and the relationship or the trend of time series data has not yet been studied [7,10,12].

In this paper, we try to analyze the trends of the time series data on the number of faults detected by unit test, integration test, function test, and validation test, and clarify the relationship between the trends and the field quality of software products. This result will make it possible to estimate field quality during testing, and make a remarkable progress in the software process improvement [1,9,13].

First, we introduce the Kendall's rank correlation coefficient $\tau$ [6] in order to discuss the trends. Based on the experimental evaluation of actual project data using the coefficient $\tau$, we define four types of trends: strict decreasing, almost decreasing, almost increasing, and strict increasing.

Then, we investigate, for each type of trend, the field quality of software products developed by the corresponding software projects. As a result of statistical analyses, we showed that software projects having trend of almost or strict decreasing in the number of faults detected by the software testing could produce the software products with high quality.

The rest of this paper is organized as follows: Section 2 describes preliminaries of this research. Section 3 shows the definition of the metrics and the overview of collected time series data to be used for the analysis. In Section 4, we classify the projects into four groups based on the trend of the number of the faults detected in the testing process. We then show the relationship between the trends and field quality in Section 5. Finally, Section 6 concludes this paper and show some future works.

## 2 Target Projects

### 2.1 Process Model

In the target projects, the products are developed under a development process as shown in Figure 1. The development process is an ordinal waterfall model. This process

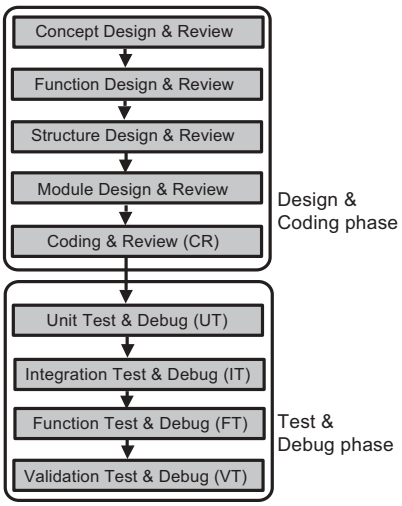



**Figure 1. Development process**

consists of two successive phases, design & coding phase and test & debug phase. The design & coding phase is divided into five activities: Concept design, Function design, Structure design, Module design, and Coding($CR$). The test & debug phase is divided into four activities: Unit test & debug($UT$), Integration test & debug($IT$), Function test & debug($FT$), and Validation test & debug($VT$).

One characteristic of the design & coding phase is that review activity is introduced after each design activity. Review activity enables the detection and correction of faults 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 to reduce their cost of producing software [2, 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 manager (this review activity is called peer review [2]). They established several guidelines for the review activity. One of them directs that at least 15% of the total efforts for design & coding phase should be assigned to review activities [9].

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 [4]. The persons engaged in the test and debug phase are directed to record all faults 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 Characteristics of Projects

The projects targeted in this paper are the development of computer control systems with embedded software in a certain company. The software products developed by the projects have the following common characteristics. The systems are related to retail systems, and thus 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 faults after delivery is very expensive. Thus, high quality is especially required for the embedded software.

We use the actual project data of 134 projects, which have already finished their development. Each project was carried out between 1995 and 1998. Additionally, we select these projects under a condition that the total number of faults detected during the test & debug phase exceeds more than a certain number. It implies that the size of target project is relatively large.

## 3 Time Series Data

### 3.1 Software Metrics

In this subsection, we define several software metrics used in this paper.

(1) Frequency of detected faults: $DF$

The frequency of detected faults ($DF$) is the number of detected faults normalized by the effort needed for an activity for detection. This metric represents the ability of developer and density of faults indirectly. In order to define $DF$, we introduce the following two parameters where $\alpha$ is an activity out of five activities $CR$, $UT$, $IT$, $FT$ and $VT$.

$D_\alpha$**:** the number of detected faults in an activity $\alpha$.

$E_\alpha$**:** the effort needed for an activity $\alpha$ (measured by person-day).

For example, the number of detected faults in the Unit test & debug activity ($UT$) is described as $D_{UT}$. Then $DF$ for the activity $\alpha$ is defined as follows:

$$DF_\alpha = \frac{D_\alpha}{E_\alpha} \tag{1}$$

According to this definition, we define the frequency of detected faults $DF_{CR}$, $DF_{UT}$, $DF_{IT}$, $DF_{FT}$, $DF_{VT}$ for activities $CR$, $UT$, $IT$, $FT$, and $VT$, respectively.

(2) Size of product: $S$

Here, we introduce the parameter $LOC$ that counts the total lines of source codes except for comment lines. Using $LOC$, the size of product $S$ is defined as follows:

$$S = \frac{LOC}{1000} \tag{2}$$

The unit of $S$ is usually represented by $KLOC$.

**Table 1. Actual project data**

| No. | $DF_{CR}$ | $DF_{UT}$ | $DF_{IT}$ | $DF_{FT}$ | $DF_{VT}$ |
|---|---|---|---|---|---|
| 1 | 2.1 | 0.6 | 0.7 | 1.8 | 0.5 |
| 2 | 6.7 | 0.0 | 1.7 | 1.2 | 0.0 |
| ... | ... | ... | ... | ... | ... |
| 133 | 11.6 | 1.8 | 0.2 | 1.0 | 2.9 |
| 134 | 3.3 | 5.3 | 15.2 | 3.3 | 6.3 |
| Average | 7.3 | 2.2 | 1.4 | 0.7 | 0.8 |
| Median | 5.2 | 1.1 | 1.1 | 0.5 | 0.3 |

(3) Field fault density: $FFD$

Field fault density $FFD$ is a metric to measure the quality of final products. It is defined as follows:

$$FFD = \frac{D_{field}}{S} \tag{3}$$

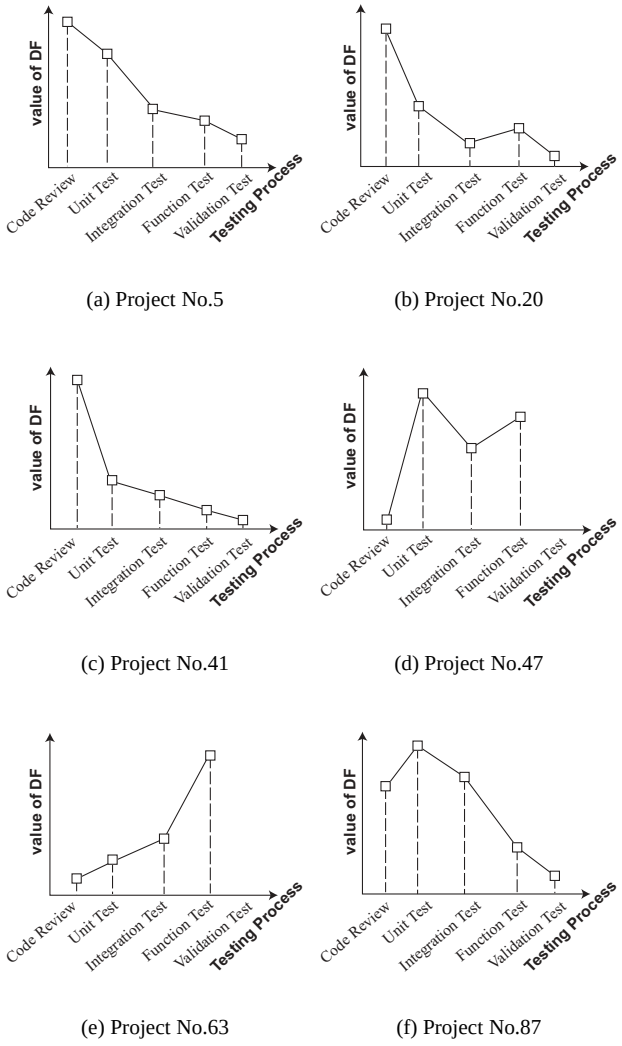
In this paper, we assume that $D_{field}$ is the number of faults detected during six months after delivery.

### 3.2 Collected Data

The effort data and faults data are recorded manually and stored in workstations by each developer. Then, they are collected by the project leader, and validated by the manager. On the other hand, field faults data are reported by a quality assurance staff, translated into a fault-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 them and reports the analysis report back to the project team and development organization [12].

Table 1 shows the actual project data. It includes quality data for $DF_{CR}$, $DF_{UT}$, $DF_{IT}$, $DF_{FT}$, $DF_{VT}$ for 134 projects. Concerning $FFD$, we have actual data, but we cannot show them here by the contract with the company. In Table 1, for projects No.1, No.2, No.133, and No.134, all metrics data are shown. However actually, for several projects, some metric data are missing. In this paper, we can use all of 134 projects since the analysis is applicable in Sections 4 and 5.

Table 1 also summarizes the average and the median of $DF$. As far as we observe the average and the median values, the values of $DF$'s are decreasing as the testing activities proceed from $CR$ to $VT$. Then, Figure 2 shows the actual values of $DF$'s for six projects. From this figure, we can find a certain trend in the successive values of $DF_{CR}$, $DF_{UT}$, $DF_{IT}$, $DF_{FT}$, and $DF_{VT}$. Intuitively speaking, projects No.5, No.20, and No.41 show the trend of decreasing in these successive $DF$'s and project No.63 shows the trend of increasing in these successive $DF$'s.




(a) Project No.5 (b) Project No.20

(c) Project No.41 (d) Project No.47

(e) Project No.63 (f) Project No.87

**Figure 2. Trend of actual data**

## 4 Classification by Trend of Faults Detection

### 4.1 Key Idea

As is shown in Table 1 and Figure 2, the projects could be classified into several groups by the trend of faults detection (for example, the trend of increasing or decreasing in the successive $DF$ values). In order to identify the trend, we introduce a quantitative measure of the trend. Although there are many metrics for measuring trend, the rank correlation coefficient is one of the most popular metrics for monotonic increasing or decreasing trend [6, 11]. In this paper, we adopt the Kendall's rank correlation coefficient $\tau$.

The following shows the definition of Kendall's rank correlation coefficient $\tau$: Here, we use two variables $X$ and $Y$ for comparison, and represent $i$-th observation by

$X_i$ and $Y_i$. For any sample of $n$ observations, there are $n(n-1)/2$ possible comparisons of points $(X_i, Y_i)$ and $(X_j, Y_j)$ $(1 \leq i, j \leq n)$. Firstly, we define $C$ to be the number of pairs that are concordant, and $D$ to be the number of pairs that are not concordant, respectively. Then, $\tau$ is defined by the following formula:

$$\tau = \frac{(C-D)}{\sqrt{[\binom{n}{2} - n_x] \cdot [\binom{n}{2} - n_y]}} \quad (4)$$

where $n_x$ and $n_y$ is the number of ties involving $X$ and $Y$ respectively. Thus, the range of $\tau$ becomes $-1 \leq \tau \leq 1$.

By using the Kendall's $\tau$ calculated by Equation 4, we can quantitatively measure the trend of faults detection. In this paper, we define the following four types of trends based on the value of $\tau$.

(a) Strict decreasing type($SD$) ($\tau = -1$)

(b) Almost decreasing type($AD$) ($-1 < \tau < 0$)

(c) Almost increasing type($AI$) ($0 \leq \tau < 1$)

(d) Strict increasing type($SI$) ($\tau = 1$)

Here, we explain the motivation of the classification for $\tau$. We firstly tried to define the two trends: decreasing ($-1 \leq \tau < 0$ ) and increasing ($0 \leq \tau \leq 1$). However, based on the experience of the actual developers, there may be a significant difference between the projects with $\tau = -1$ (or $\tau = 1$) and $-1 < \tau < 0$ (or $0 \leq \tau < 1$). We thus use the above four types in this paper.

Projects No.5 and No.41 with $\tau = -1$ in Figures 2(a) and 2(c), respectively, are classified into type $SD$. Then, project No.20 with $\tau = -0.8$ in Figure 2(b) and project No.87 with $\tau = -0.6$ in Figure 2(f) are type $AD$. Project No.47 with $\tau = 0.33$ in Figure 2(d) is type $AI$. Finally, project No.63 with $\tau = 1$ in Figure 2(e) is type $SI$. This result implies that the value of Kendall's $\tau$ is adequate for distinguishing these trends.

The followings present interpretations of four trends:

(a) Strict decreasing: Many faults are detected and removed in the early stage, and then a few faults are detected and removed in the later stage. Thus, it is expectable that only a very few faults remain in the final software product.

(b) Almost decreasing: This type is similar to "strict decreasing type". But the trend is not strict, and thus a few remaining faults may be found.

(c) Strict increasing: A few faults are detected and removed in the early stage, and then many faults are detected and removed in the later stage. Thus, many faults still remain in the final software product.

(d) Almost increasing: This type is similar to "strict increasing type". But the trend is not so strict, and thus many faults may remain.

**Table 2. Classification by fault detection trend**

| Types | Number of projects (%) |
|---|---|
| Strict decreasing (SD) | 51 (38.1%) |
| Almost decreasing (AD) | 64 (47.8%) |
| Almost increasing (AI) | 16 (11.9%) |
| Strict increasing (SI) | 3 (2.2%) |

Because of taking these trend for classification, the effect of the factors such as the skill level of development team, the kind of product, etc. is mitigated and included in the trend.

### 4.2 Classification Result

Table 2 shows the result of classification by applying Kendall's rank correlation coefficient $\tau$ to 134 projects. We can see that almost $38\%$ projects are in type $SD$, and almost $48\%$ projects are in type $AD$. Thus, $86\%$ projects have decreasing trends. But the rest $14\%$ projects have increasing trends (that is, types $AI$ and $SI$).

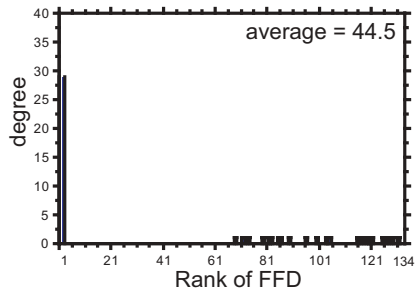
In order to find the relationship between the $FFD$ and four types, we present histograms of the $FFD$ in Figure 3. Please note that y-axis of Figure 3 denotes the ranks of the values of $FFD$ rather than the values themselves. Here, the rank of each project takes a value from 1 to 134. Thus, rank$= 1$ implies the highest quality (that is, $FFD = 0$ in actual data) and rank$= 134$ implies the lowest quality (for this case, we cannot show actual value of $FFD$ by the contract). Figures 3(a) and (b) show that almost half of projects with types $SD$ and $AD$ have the rank$= 1$ and thus have very high quality. On the other hand, Figure 3(c) and (d) show that types $SI$ and $AI$ tend to include projects with lower quality on the average. Figure 3 also shows the average rank of the field quality for each type.

From Table 2 and Figure 3, we can expect that the average field quality of the projects with the types $SD$ and $AD$ is rather good. On the other hand, the field quality is not so good for types $SI$ and $AI$. Concerning this property, we will investigate further in the next section.

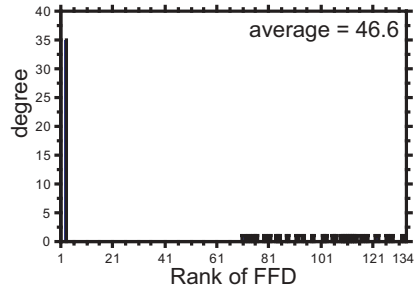
## 5 Relationship between Trend and Field Quality

### 5.1 Outline of Analysis

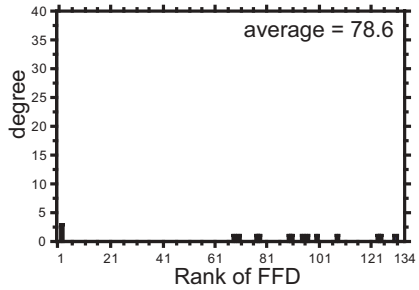
In order to compare field quality of projects having four types of trends, we introduce the parameter $\theta$ of the location on the rank of the $FFD$. In this paper, we define $\theta_{SD}$, for projects with type $SD$, to be the average rank of $FFD$. Similarly, we define $\theta_{AD}$, $\theta_{AI}$, and $\theta_{SI}$ for projects with



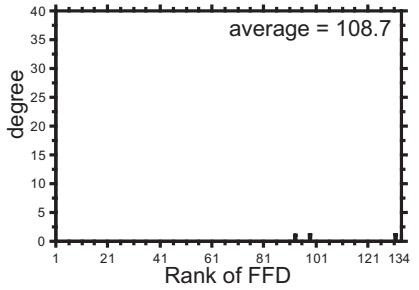

(a) Strict decreasing type




(b) Almost decreasing type




(c) Almost increasing type




(d) Strict increasing type

**Figure 3. Histogram of $FFD$ for each type**

types $AD$, $AI$, and $SI$, respectively. For simplicity, we call $\theta_{SD}$, $\theta_{AD}$, $\theta_{AI}$, and $\theta_{SI}$ the average $FFD$ of projects with types $SD$, $AD$, $AI$, and $SI$, respectively. Then, from the interpretations, we naturally derive the following relationship.

$$\theta_{SD} \leq \theta_{AD} \leq \theta_{AI} \leq \theta_{SI} \tag{5}$$

In this paper, we planned two steps of analysis for the average $FFD$, $\theta_{SD}$, $\theta_{AD}$, $\theta_{AI}$, and $\theta_{SI}$ of projects.

Step 1. (Jonckheere Test [8]) We check if there is a significance difference among $\theta_{SD}$, $\theta_{AD}$, $\theta_{AI}$, and $\theta_{SI}$ and there exists the order shown in Equation 5.

Step 2. (Multiple Comparison) Since the order exists among $\theta_{SD}$, $\theta_{AD}$, $\theta_{AI}$, and $\theta_{SI}$, we perform Ryan's procedure [5]. Ryan's procedure consists of the following procedure: Firstly, compare the farthermost pair on order. Next, compare the next farthermost pairs. By performing pairwise comparison step by step, we finally analyze a significant difference between neighboring pairs.

## 5.2 Experimental Evaluation

**(A) Jonckheere test**

We define two hypotheses $H_0$ and $H_1$ for the average $FFD$, $\theta_{SD}$, $\theta_{AD}$, $\theta_{AI}$, and $\theta_{SI}$ of projects. The hypothesis $H_1$ is clearly alternative of the null hypothesis $H_0$. In this analysis, the level of significance $\alpha$ is chosen as $0.05$.
$H_0$: There is no difference among $\theta_{SD}$, $\theta_{AD}$, $\theta_{AI}$, and $\theta_{SI}$. That is, the following relation holds:

$$\theta_{SD} = \theta_{AD} = \theta_{AI} = \theta_{SI} \tag{6}$$

$H_1$: Either of the following relations comes true:

$$\theta_{SD} < \theta_{AD} \leq \theta_{AI} \leq \theta_{SI} \tag{7}$$
$$\theta_{SD} \leq \theta_{AD} < \theta_{AI} \leq \theta_{SI} \tag{8}$$
$$\theta_{SD} \leq \theta_{AD} \leq \theta_{AI} < \theta_{SI} \tag{9}$$
$$\theta_{SD} < \theta_{AD} < \theta_{AI} \leq \theta_{SI} \tag{10}$$
$$\theta_{SD} < \theta_{AD} \leq \theta_{AI} < \theta_{SI} \tag{11}$$
$$\theta_{SD} \leq \theta_{AD} < \theta_{AI} < \theta_{SI} \tag{12}$$
$$\theta_{SD} < \theta_{AD} < \theta_{AI} < \theta_{SI} \tag{13}$$

By Jonckheere test, since the probability that $H_0$ cannot be rejected becomes 0.02, the null hypothesis $H_0$ is rejected at the $0.05$ level. Thus, the alternative hypothesis $H_1$ is accepted. This result implies that there is a statistically significant difference in the average $FFD$, $\theta_{SD}$, $\theta_{AD}$, $\theta_{AI}$, and $\theta_{SI}$ of projects as a whole, and the field quality becomes larger in the order of $SD$, $AD$, $AI$, and $SI$. However, at this stage we cannot know which relation in the hypothesis $H_1$ holds.

**Table 3. Result of Ryan's pairwise comparison**

| Step | Nominal significance level $\alpha'$ | Pair of types | p-value |
|---|---|---|---|
| 1 | $\alpha'$=0.017 | ($\theta_{SD}$, $\theta_{SI}$) | 0.017 |
| 2 | $\alpha'$=0.025 | ($\theta_{SD}$, $\theta_{AI}$) | 0.016 |
| | | ($\theta_{AD}$, $\theta_{SI}$) | 0.024 |
| 3 | $\alpha'$=0.05 | ($\theta_{SD}$, $\theta_{AD}$) | 0.430 |
| | | ($\theta_{AD}$, $\theta_{AI}$) | 0.026 |
| | | ($\theta_{AI}$, $\theta_{SI}$) | 0.089 |

**(B) Ryan's multiple comparison**

Next, the result of Ryan's pairwise comparison is summarized in Table 3. Step 1 shows there is somewhat significant difference between $\theta_{SD}$ and $\theta_{SI}$. Similarly Step 2 shows there is a significance difference between $\theta_{SD}$ and $\theta_{AI}$, and also between $\theta_{AD}$ and $\theta_{SI}$. Step 3 implies that there is significant difference between $\theta_{AD}$ and $\theta_{AI}$. But, there is not significance difference between $\theta_{SD}$ and $\theta_{AD}$, and between $\theta_{AI}$ and $\theta_{SI}$. One of the reasons why there is no difference between type $AI$ and $SI$ is that the number of projects included in the class $SI$ was rather small. We therefore need more project data and have to perform further analysis as a future work.

From these facts, Equation 8 holds for the average $FFD$ of projects. This implies that the average rank of the field fault density $FFD$ becomes larger according to the order of $\theta_{SD}$, $\theta_{AD}$, $\theta_{AI}$, and $\theta_{SI}$, and that especially large difference exists between $\theta_{AD}$ and $\theta_{AI}$.

In subsections 4.1 and 4.2, we estimated that two types $SD$ and $AD$ behave the same way with respect to (the rank of) $FFD$. Similarly two types $SI$ and $AI$ behave the same way. But, there is a large difference between two types $AI$ and $AD$. The estimation agrees with this result.

## 6 Conclusion

In this paper, we have analyzed time series data on the number of faults detected by successive code review and testing activities. By applying the rank correlation coefficient to actual project data, we have successfully classified the data into four types of trends: $SI$, $AI$, $AD$ and $SD$. Then, we have investigated the relationships between trends and field quality, and showed that software project having trend $AD$ or $SD$ would produce high quality products.

However, since we used only the data from embedded software projects, we have to analyze the data from other kinds of projects. On the other hand, construction of a model, which calculates the number of remaining faults based on time series quality data, is also an important future work.

The goal of this research is to estimate field quality based on the time series data on the number of faults detected by various review activities and testing activities. In order to attain the goal, we must analyze the reason why the target project has a specific trend. Especially, for the trends $SI$ and $AI$, we need a process improvement based on the reasons.

## References


[1] V. Basili and J. Musa. The future engineering of software: A management perspective. *IEEE Computer*, 24(9):90–96, 1991.

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

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

[4] International Standard Organization. *IEEE standard glossary of software engineering terminology,*. IEEE Std 610.12-1990, 1990.

[5] S. Iwahara. *Psycological statistics: Non-parametric method*. Nihon Bunka Kagakusha Co.,Ltd. (in Japanese), 2 edition, 1968.

[6] M. Kendall and J. D. Gibbons. *Rank correlation methods*. Edward Arnold, 5 edition, 1990.

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

[8] E. L. Lehmann. *Nonparametrics: Statistical methods based on ranks*. Holden-Day, Inc., 1975.

[9] O. Mizuno and T. Kikuno. Empirical evaluation of review process improvement activities with respect to post-release failure. In *Empirical Studies on Software Development Engineering (ICSE'99 Workshop)*, pages 50–53, 1999.

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

[11] S. Muto. *Statistical analysis handbook*. Asakura Books (in Japanese), 1995.

[12] 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 6th Int'l Symposium on Software Reliability Engineering(ISSRE'95)*, pages 34–39, 1995.

[13] T. Tanaka, K. Sakamoto, S. Kusumoto, and K. Matsumoto. Improvement of software process by process description and benefit estimation. In *Proc. of 17th International Conference on Software Engineering(ICSE'95)*, pages 123–132, 1995.