Improvement of the Fault-prone class prediction precision by the process metrics use

Nobuko Koketsu
NEC
Tokyo, Japan
n-koketsu@bq.jp.nec.com

Naomi Honda
NEC
Tokyo, Japan
n-honda@ay.jp.nec.com

Shinya Kawamura
NEC
Tokyo, Japan
s-kawamura@bl.jp.nec.com

Jyunichi Nomura
NEC
Tokyo, Japan
j-nomura@aj.jp.nec.com

Makoto Nonaka
Toyo University
Tokyo, Japan
Nonaka-m@toyo.jp

Abstract

The general purpose of our study is to introduce a more precise and effective quality assurance practice into our software development organization that has been applying quality assurance activities mainly focusing on process metrics. The fault-prone (FP) module prediction technique is one of promising techniques that meets our goal. In this paper, we explain our attempt to build several FP class prediction models to predict defects that would be detected in a functional testing phase by using our actual product development data. We also discuss practical consideration of how to apply the FP technique to a testing phase in practice to stabilize software quality earlier than before.

1. Introduction

Both our customers and society as a whole are demanding reliability from ever-larger and more complicated software products. At the same time, we have to provide the market with the software features that it requires, in both a timely manner and at a reasonable cost. These trends therefore demand a technology that can efficiently ensure a higher level of reliability. A fault-prone (FP) module prediction technique is expected to be one of the effective solutions that satisfy these technological needs.

The goal of a fault-prone (FP) module prediction technique is to identify any module that is more likely to contain a fault, from among all the modules constituting the software. If the FP module predictions can be made accurately, testing strategies based on the results of those predictions can be planned and implemented. This is expected to reduce quality assurance costs. Many studies of FP module prediction have been conducted to date. Initial research started in the 1980s[1] and evolved, during the 1990s, into studies into prediction techniques [3][4][5] that use Chidamber & Kemerer (CK) metrics as an explanatory variable for object-oriented software. Since 2000, the focus of studies has moved to open source software (OSS) [6][7][8][9], as well as the application of various prediction models[6][8][10], excluding the dominant multiple logistic regression model used in the past. The FP module prediction studies are becoming widespread.

It is hard to say, however, whether the organizations or data contexts to be analyzed for these FP module prediction studies are similar to those being investigated by the authors. Our organization has continued its activity for software quality improvement [11] since the 1970s, long implementing quality assurance activities that are centered on a technique known as the “accounting system” [12]. In 2004, our organization attained a CMMI level of 5. This indicates that the maturity level of our organization's software life cycle process is relatively high. Considering advanced research into the FP module prediction technique based on these circumstances, we can point out those issues that arise from the application of the technique to the organization. For instance, the organization examined in the research by Basili et al.[13] is very different from our organization in the context of process maturity levels, due to the fact that the students used data from projects under development. Zhou et al. horizontally investigated FP module prediction studies using CK metrics and concluded that the metrics for class scale and coupling almost always had a degree of statistical significance [5]. However, it is necessary to fully consider the ways in which the contexts differ.

The authors reviewed the adoption of the FP module prediction technique as a means of enhancing the precision of quality assurance, adding the technique to the traditional quality assurance system that depends on accumulated metrics. The FP module prediction in this paper identifies classes with modules. In this paper, therefore, we will subsequently refer to FP module prediction as FP class prediction. For this study, we tried to construct and evaluate our own FP class prediction model equation using the metrics collected during actual development, in consideration of the findings and issues related to the advanced studies. As a result, an FP class prediction model equation, which appears to be effective, could be constructed by combining the complexity levels, CK metrics and other product metrics data with the conventional process metrics, through the use of a Bayesian network model. Furthermore, the established FP class prediction was conducted upon the completion of the coding step to review a method of quickly stabilizing the software quality in the testing phase. This paper describes the model, the method, and the processes.

2. Outline of Development Products and Processes

This section outlines the products to be developed by our organization, as well as the product development and quality assurance processes. Our organization develops and maintains operating systems and middleware products. These products are designed for general-purpose applications, not for specific customers. Consequently, if a software product gives rise to any quality problems post-release, the impact can be far-reaching. For this reason, a higher level of quality is required of the product. When developing a new software product, it is normal for an existing product to be repeatedly updated to enhance its features.

2.1 Development Processes

The Process phase of software development is defined as being the product development and quality improvement process. The product development process is based on the V model shown in Figure **1**. This model consists of an upstream process of four phases, from basic design to coding, and a testing process of three phases, from the unit test to the system test. Each phase defines the deliverables, implementation tasks, and measurement items for the individual steps. Of the measurement items, typical process metrics are listed in Table 1. The quality improvement process is systematically defined for all the phases in the product development process, or the activities to be undertaken to prevent faults and the recurrence of those faults. These development processes are standardized in an organized manner and uniformly applied to the development of a product by the development department.

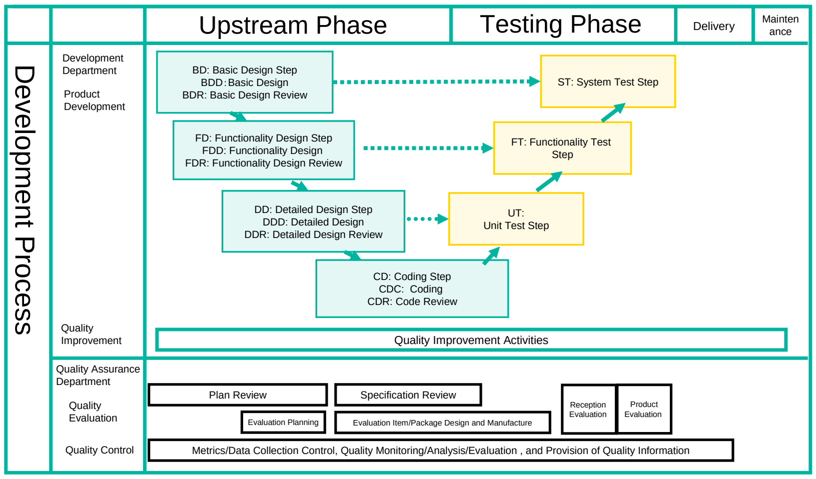


Figure 1. Development process and quality assurance activities

2.2 Quality Assurance Processes

The goal of the quality assurance processes is to assure the qualities of both the deliverables prepared by, and the tasks performed by, the development department. These processes are performed independently by the quality assurance department. The quality assurance processes are defined as being the quality evaluation and quality control processes. Independent evaluations such as the validation of the development plan and the specification are done for the quality evaluation process. In the quality control process, the data shown in Table 1 is collected and measured at every phase of the development processes. Based on this data, the quality and the progress of the development of a product are determined objectively and multilaterally to identify any issues.

Table 1. Typical process metrics

Phases to be measured	Metrics
Entire development processes	Starting date delay (days), completion date delay (days), defect count/KL, and defect detection rate
Only upstream process	Rate of work progress, effort/KL, review effort/KL, and defect count/review effort
Only testing process	Execution ratio of test items, number of test items/KL, and testing effort/KL

3. Method of Applying FP Class Prediction Model to Product Development

In the actual development of a product, the product is divided into several subsystems that each correspond to a functional unit. The quality control focuses on the collected metrics. The control that is based on the metrics is basically conducted according to the data that is collected for the individual subsystems.The quality of the entire subsystem could only be determined numerically. Quality control with a higher precision than this was left to individual quality analyses that were done by each developer. In the testing phase, the number of remaining faults was controlled under the accounting system, and control was also applied in units of subsystems. If an FP prediction model equation is constructed with product metrics collected from the source code, it can be established using finer units than subsystems. For instance, an FP prediction based on each class enables quality analysis on 215 class units, although the development was traditionally controlled based on 7 subsystem units. Classes that are likely to cause any modification to the steps subsequent to the functionality test are predicted at an early stage of the development, and a higher priority is given to testing a class that is determined to be FP. We thought that the implementation of such a test would lead to the earlier stabilization of the quality. Hence, we tried to construct an FP class prediction model equation using actual development data.

4. Outline of Data for Analysis

4.1 Data for Analysis

In this paper, we consider the analysis of a general-purpose middleware product. We analyzed the data used for the initial development of a new product and for the development required for the first update. In the development processes of the development for updating the product, a method for separating the phases was roughly divided into two ways according to its subsystems. As this was considered to have a reasonable impact on the process metrics data, the subsystems that were developed as part of the phases for developing the new product were classified as B, and those that were not were classified as C, for each data set. The development uses mainly C and C++, but we analyzed only those portions written in C++. An outline of the data sets used for the analysis is given in Table 2.

Table 2. Outline of data used for analysis

Reference symbol	Development content	Development scale	Analysis scale	Number of classes	Number of subsystems
A	New development	105KL	57KL	215	7
B	Updating development 1	177KL	165KL	611	9
C	Updating development 2	124KL	91KL	450	6

4.2 Metrics for Analysis

In addition to the metrics listed in Table 1, the per-class product metrics that were collected for the FP module predictions are listed in Table 3. Of the metrics in Table 3, the CK metrics were measured using TechMatrix Corporation's Understand [15]. Other metrics, which are continuously being collected as part of the quality control process mentioned in Section 2.2, were measured with a tool that we developed in-house.

Table 3. Per-class product metrics

Type	Metrics	Outline
Scale	Number of effective lines	Value derived by subtracting the comment and blank lines from the total number of lines (total summation per class)
	Number of methods	Number of methods contained in class
Complexity level	Cyclomatic complexity	Value representing route complexity by branching command (total summation per class)
	Number of branch conditions	Number of conditional equations for branching command (total summation per class)
	Maximum number of nesting levels	Class average of maximum number of nesting levels for each method
CK	WMC	Total of weighted number of methods
	DIT	Number of hierarchies up to root class in inheritance tree
	NOC	Nmber of direct subclasses
	CBO	Number of other classes that refer to and execute methods and instance variables
	RFC	Total of methods in which an object is executed in response to received messages
	LCOM	Number of methods in which common attributes are manipulated, which represents a lack of cohesion.

5. Fault-Prone Class Prediction

A to C in Table 2 are used for sample data. Here, 1 (with fault) and 0 (without fault) were assigned to classes in which any fault was observed after the functionality test and to those in which it was not, respectively, for response variables. Nine evaluation patterns can be considered, depending on which data set is used for the creation and evaluation of a model. The different patterns are listed in Table 4.

Table 4. FP class prediction evaluation patterns

Reference symbol	P1	P2	P3	P4	P5	P6	P7	P8	P9
Data for model creation	A	B	C	A	A	B	B	C	C
Data for model evaluation	A	B	C	B	C	A	C	A	B

5.1 Evaluation Indices for FP Predictions

The evaluation indices used for the FP class predictions are listed in Table 5. An F value and precision ratio from among the five indices given in the literature [8], as well as our original index were used as an evaluation index. The F value, which is given by Equation (1), is often utilized as an evaluation index for FP predictions because the predictions can be generally evaluated while maintaining a balance between the recall and precision ratios representing a tradeoff relationship.

$$F\ value = \frac{2 \times Recall \times Precision}{Recall + Precision} \qquad (1)$$

Our original index, or an early fault class test ratio, is an indicator that represents the degree that can early test a faulty class when a test is conducted in the order of "class determined to be FP" and "class determined not to be FP," according to the results of FP class prediction. More specifically, the execution ratio of the test items subsequent to the functionality test is placed on the horizontal axis, and the number of detected faults is placed on the vertical axis. Next, the detection of faults with the progress of the test is plotted as a graph. The area surrounded by this graph and the horizontal axis is used to evaluate the precision of the prediction model (these values were established in reference to the AUC evaluation by Kaur et al.). Empirically, some degree of efficacy is observed with a value of 0.55 or more. A value of 0.65 or more can be judged to be very effective. A higher-precision prediction model will increase the number of faults detected relatively quickly, which is expected to stabilize the quality of the product at an early stage.

Table 5. Evaluation indices for FP class predictions

Evaluation indices	
Recall ratio	Ratio of modules correctly determined to be FP among those modules that were actually faulty
Precision ratio	Ratio of modules that were actually faulty among those modules determined to be FP
F value	Harmonic mean of recall and precision ratios. A larger harmonic mean represents a higher-precision determination.
Early fault class test ratio	Original index. Average of the rate determined not to be FP among all classes, and a recall ratio.

The purpose of this FP class prediction is to stabilize the quality more quickly by testing the predicted classes at an early stage. For this reason, the early fault class test ratio was treated as the most important index, while the F value was treated as the second main index. To this end, a threshold for ensuring the efficacy of the prediction model was established. If the ratio is less than 0.55, or if the F value is less than 0.4 (these values were established in reference to the studies by Kamei et al. [6]), the prediction model shall be rejected due to its nonconformity.

5.2 Creation and Evaluation of Logistic Regression Model

First, a multiple logistic regression model was used to construct and evaluate a model. Evaluation results with high precision were not obtained when the model was evaluated with data other than that used for the Creation. In general, a linear regression model assumes the absence of a mutually dependent relationship between explanatory variables. On the other hand, due to a strong mutually dependent relationship between the explanatory variables in the product and process metrics of the software, it is assumed to be difficult to construct a highly robust prediction model.

5.3 Creation and Evaluation of Bayesian Network Model

The linear regression model did not provide a highly robust prediction model. Hence, a network-type model assuming a mutually dependent relationship between explanatory variables was used to try to predict FP classes with a Bayesian network. The Bayesian network is one type of network model that stochastically describes cause and effect relationships, or a probabilistic inference model that expresses inferences for relationships based on a directed graph, and individual variable relationships based on a conditional probability.

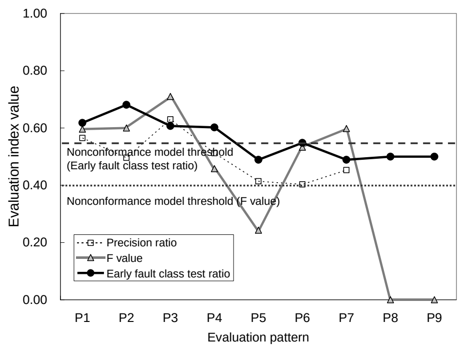
This paper evaluates three types of Bayesian networks. The characteristics of the networks are listed in Table 6. BN1's NaiveBayes assumes the absence of a mutually dependent relationship between the explanatory variables, but was evaluated given that it is the most commonly used Bayesian network model. BN2's TAN introduces a mutually dependent relationship between explanatory variables into the NaiveBayes. BN3's Bayes Net has the highest degree of network flexibility. A maximum value that is described based on the mutually dependent relationship between explanatory variables in BN2 and BN3 refers to the maximum number of explanatory variables that one explanatory variable depends on.

Table 6. Characteristics of Bayesian network models to be evaluated

Reference symbol	Model type name	Mutually dependent relationship between objective and explanatory variables	Mutually dependent relationship between explanatory variables
BN1	NaiveBayes	Required	Absence
BN2	TAN (Tree Augmented NaiveBayes)	Required	Presence (Max 1)
BN3	Bayes Net	Optional	Presence (Max 3)

To evaluate the precision of the prediction of the Bayesian network models in consideration of robustness, a bootstrap method was used to evaluate the models. More specifically, 80% of all the samples were randomly detected from model application data to construct a model. Then, 4000 iterations were made to evaluate the constructed model.. The average obtained from these iterations was treated as the final predictive value. In the same pattern of model creation and evaluation data (P1, P2, and P3), cross-validation and the bootstrap method were used concurrently.

The results of the evaluation are shown in Figure 2,Figure 3, and Figure 4 When we compare, Figure 2,Figure 3, , and Figure 4, We find that TAN in Figure 3 exceeds the other models in terms of a small number of nonconformance models, early fault class test ratios and F values, providing the most effective prediction results. In particular, in the light of the absence of nonconformance models, we can assume that a prediction model with a certain level of robustness could be ensured. When prediction models were constructed using the bootstrap method, the models lacked stability whenever an iteration was made.In the results, a prediction model with a fixed explanatory variable could not be identified. If the number of data pieces for model creation is increased in the future, a high-precision prediction model may be constructed even with other methods, such as Bayes Net



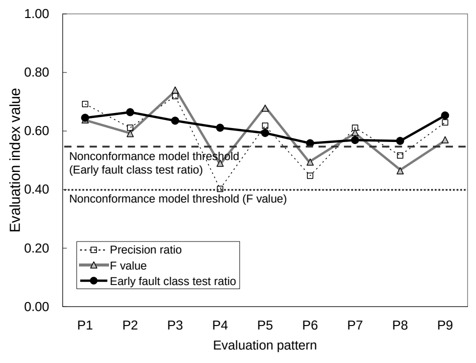
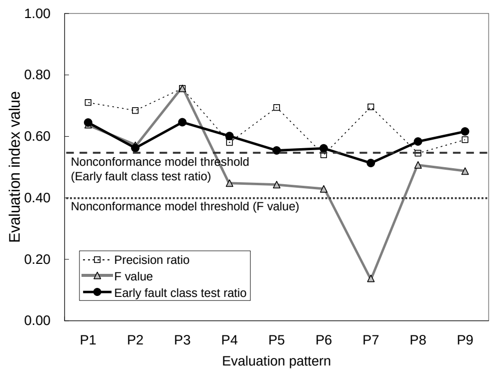


Figure 2. Results of evaluation of NaiveBayes



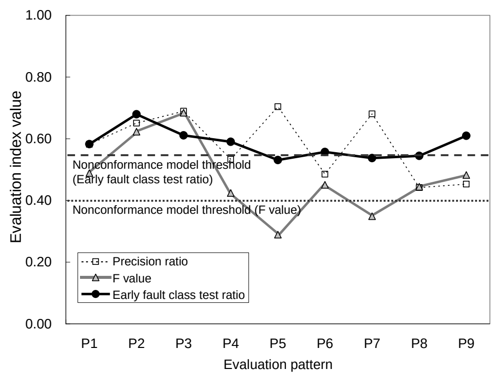


Figure 3. Results of evaluation of TAN

Figure 4. Results of evaluation of Bayes Net

Figure 5. Results of evaluation of TAN (no process metrics)

To verify whether process metrics contribute to improving the prediction precision in the model of TAN, data in which the process metrics are excluded from the explanatory variables was used to construct and evaluate the model. The results of the evaluation are shown in Figure 5. In the comparison of Figure 3 and Figure 5, the number of nonconformance models in Figure 5 is larger than in Figure 3, and the early fault class test ratios and F values in Figure 5 are lower than those in Figure 3 indicates that the process metrics contribute to the increase in the TAN model prediction precision.

6. Application of the Model to Actual Development

The efficacy of this model is investigated when the model is applied to the actual development of a product and a fault is predicted with the model upon the completion of the coding step. As one example, Table 7 shows the relationship between the result (presence or absence) of any detected fault in the steps subsequent to the functionality test, and the result of the prediction by the FP class prediction model constructed by TAN (Figure 3) in the pattern of P4 in Table 4.

Table 7. Relationship between the result of any detected fault and the FP class prediction

Accomplishment	**Results of FP class prediction**		**Total**
subsequent to the functionality test	**The absence of any fault**	**the presence of any fault**	
Absence of any fault in the steps	303	149	452
Presence of any fault in the steps	59	100	159
Total	362	249	611

As shown in Table 7, the fault was corrected in 159 of 611 classes in the steps subsequent to the functionality test. On the other hand, 249 classes exhibited the possibility of the occurrence of a fault. In 100 of the 249 classes, the fault was actually corrected. Assuming that the test is conducted uniformly in all the classes and that the probability of the occurrence of the fault in the test items is uniform, 63% of the test items (100/159) must be executed after the functionality test in order to detect faults in 100 classes. If the test is conducted for a class that is estimated to have a high possibility of the occurrence of a fault under the FP class prediction, it will become possible to detect the fault in 100 classes when 41% of all the test items

(249/611) are completed, thereby improving the quality of the product at an early stage of the test. In reality, the pace of detection of faults is not constant as the test progresses, but in any way, the trial discussed in this paper is regarded as being more likely to contribute to the early stabilization of the quality. Furthermore, a class that has a high possibility of having a fault remaining upon the completion of the coding step is identified, thereby enabling the organization to take measures to ensure quality, such as examining the need of review or increasing the coverage of unit tests, in the steps leading up to the functionality test and resulting in the realization of faster quality stabilization.

7. Conclusion

For this study, we constructed and evaluated FP class prediction models using product and process metrics. We could not identify effective explanatory variables for predicting a faulty class after the functionality test, but could establish an effective prediction model with some robustness by constructing Bayesian network models with a combination of product and process metrics. In this study, a model equation was constructed that predicted the correction of faults detected in the steps subsequent to the functionality test. The faults include the corrections required by specification changes and the addition of features. These corrections cannot be measured with source codes and class structures. Consequently, it is necessary to develop the model equation to take into account causes that generate faults, as well as the classification of those faults, in order to enhance the precision of the equation.

When the actual use of the model equation is assumed, if the equation identifies the quality per class, it may be more effective in improving the precision of quality assurance than doing so per subsystem. In this paper, this model equation was not applied to actual development scenarios. In the future, we would like to review detailed measures for effectively applying the prediction using the model equation to actual development, as well as to identify explanatory variables.

References

[1] Shen, V., Yu, T., Thebaut, S. and Paulsen, L.: Identifying Error-Prone Software:An Empirical Study, IEEE Trans. Softw. Eng., Vol.11, No.4, pp.317–324 (1985).

[2] Chidamber, S. and Kemerer, C.: A Metrics Suite for Object Oriented Design, IEEETrans. Softw. Eng., Vol.20, pp.476–493 (1994).

[3] Basili, V.R., Briand, L.C. and Melo, W.L.: A validation of object-oriented design metrics as quality indicators, IEEE Trans. Softw. Eng., Vol.22, No.10, pp.751–761(1996).

[4] Briand, L.C., W¨ust, J., Daly, J.W. and Porter, D.V.: Exploring the relationship between design measures and software quality in object-oriented systems, J. Syst.Softw., Vol.51, No.3, pp.245–273 (2000). 347278.

[5] Zhou, Y. and Leung, H.: Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults, IEEE Trans. Softw. Eng., Vol.32, No.10, pp.771–789 (2006).

[6] Yasutaka Kamei, Shuji Morisaki, Akito Monden, Kenichi Matsumoto: Fault-prone module determination methods using combination of association rule analysis and logistic regression analysis, Information Processing Society of Japan Journal, Vol.49, No.12, pp.3954–3966 (2008).

[7] Gyimothy, T., Ferenc, R. and Siket, I.: Empirical validation of object-oriented metrics on open source software for fault prediction, IEEE Trans. Softw. Eng., Vol.31, No.10, pp.897–910 (2005).

[8] Kaur, A. and Malhotra, R.: Application of Random Forest in Predicting Fault-Prone Classes, Intl. Conf. Advanced Computer Theory and Eng. (2008).

[9] Shatnawi, R. and Li, W.: The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process, J. Syst. Softw., Vol. 81,No.11, pp.1868–1882 (2008).

[10] Menzies, T., Greenwald, J. and Frank, A.: Data Mining Static Code Attributes to Learn Defect Predictors, IEEE Trans. Softw. Eng., Vol.33, pp.2–13 (2007).

[11] Yukio Mizuno, company-wide SWQC activity adjustment committee (Eds.): Comprehensive quality control of software: NEC's SWQC activities, JUSE Press, Ltd. (1990).

[12] Naomi Honda: Software quality accounting: Quality assurance technologies supporting NEC's high-quality software development, JUSE Press, Ltd.(2010).

[13] Ohlsson, N. and Alberg, H.: Predicting fault-prone software modules in telephone switches, IEEE Trans. Softw. Eng., Vol.22, No.12, pp.886–894 (1996).

[14] Fenton, N., Neil, M., Marsh, W., Hearty, P. A. H.P., Radlinski, L. A. R.L. and Krause, P. A. K.P.: Project Data Incorporating Qualitative Facts for Improved Software Defect Prediction, Proc. 3rd Int'l. Workshop on Predictor Models in Softw.Eng. (PROMISE'07), 10 pages (2007).

[15] http://www.techmatrix.co.jp/quality/understand/