Identifying Key Attributes of Projects that Affect the Field Quality of Communication Software

Nahomi Kikuchi†‡, Osamu Mizuno‡and Tohru Kikuno‡
kikuchi386@oki.co.jp, {o-mizuno, kikuno}@ics.es.osaka-u.ac.jp
†Oki Electric Industry Co., Ltd., Japan
‡Graduate School of Engineering Science, Osaka University, Japan

Abstract

In this paper we identify key attributes of projects in which the number of problem reports after release is remarkable in the communication software. After several interviews to software project managers, we derived candidate attributes of importance to the projects. To find out the most influential attributes of the projects, we conducted statistical analysis using a set of metrics data related to the candidate attributes and the number of problem reports after release. As a result, we successfully found that two metrics concerning the origin's quality and the changes in specification are useful to estimate the field quality.

1 Objective and Approach

The quality of communication software is usually measured by the problem reports after release in the field. That is, the quality is only assessed by the resultant product. Actually there exist many studies on the quality of software products[1, 2, 3]. Generally, review and test activities are effective to acquire better quality. Some of them discussed the methodologies of review and test[1, 2], and others proposed statistical models[3] to analyze how the quality of software increases. They, however, focused on the development of general software(rather than specific domain software), and they did not consider explicitly the deadlines of projects.

The aim of this paper is to investigate the common attributes of projects whose field quality was not good. In this paper, we don't consider projects that did not finish within predetermined dates. The reason is that for such projects, no reliable data are collected during development. It is thus impossible to analyze those projects statistically.

Satisfaction of customers is then evaluated by field quality after release. In addition, quality after the release is reflected on problem reports after the release in the field. Thus, we judge success of a project using the number of problem reports originated by customers after release for one year (called problem reports for short).

We chose twenty-four projects in a company that have commonly the following characteristics:

(1) Each project was completed within the schedule.

(2) Each project used similar programming languages (for instance C, C++, etc.).

In order to find characteristics of projects, we classified these projects into two groups: Good and Fair. To find characteristics of the Fair projects, we hold several interviews with a number of project managers from both groups. Then, we examined and presented a list of candidate attributes of importance to the projects' post-release quality. To identify influential attributes of the projects, we conducted statistical analysis using a set of metrics data related to the candidate attributes and the numbers of problem reports.

2 Communication Software Development

In the development of communication software, there exist several types of software development such as reusing or adapting from existing software, a previous version of the current product, commercial off-the-shelf software (COTS), and commercial or reuse libraries. The pre-existing software, which is used as development basis, is generally called the origin[6]. We then call a software product that was developed using pre-existing origins Prior work. Other type of a software product is newly developed software without using pre-existing one, and is called New work.

Prior work is further divided into Modified, Adopted and Reuse according to development activities. "Modified" part of software is developed based on pre-existing software with many changes on design and codes. On the other hand, "Adapted" and "Reuse" part of software are developed with almost no design and code change. In "Adapted" part of software, the design is reviewed and program is tested to a limited extent. In "Reuse" part of software, the design is neither reviewed nor tested at all.

The development process is a sequential software engineering life-cycle model according to the waterfall model. Software requirement is determined in the analysis phase and development plan is constructed in the planning phase. Then implementation is performed in the design phases(basic, functional and detail), the programming phase and the test phases(unit, module and system). In each phase, software work products and documents are created and reviewed to remove defects. For the purpose of project management, development reports are written at the end of each phase.

A software project manager describes development reports and evaluates the progress and status to see if the project may proceed to the next phase.

While other related systems such as hardware and firmware are developed at the same time, the software part is sometimes asked for the change of interface specification in certain timing(for example, during the design phase). When a specification is changed, some modules may also be changed. Such changes of specification are likely to happen frequently according to the stability of related systems.

Since the software development process is usually performed concurrently with the related systems' development, it is also difficult to verify performance specifications, such as peak performance at real operational situation, in design phases. Thus, insufficient specifications suddenly appear in the test phase, and they often require changes in the design specifications.

3 Interview

3.1 Categorization of projects

In order to distinguish projects whose quality is not considered as satisfactory, we adopted the metrics Ft_1. (We introduce other related metrics for statistical analysis in Section 4.)

Ft_1 : the number of problem reports for twelve months after release

We call projects with unsatisfactory value for Ft_1 Fair project. Furthermore, we classified Fair projects into two subgroups Poor and Average based on the degree of their post-release quality. Classification rules are summarized as follows:

$$\begin{cases} Good \quad project & ; Ft_1 < n_1 \\ Fair \quad project & \begin{cases} Average \quad project & ; n_1 \leq Ft_1 < n_2 \\ Poor \quad project & ; n_2 \leq Ft_1 \end{cases} \end{cases}$$

These numbers n1 and n2 were determined by having discussion with the quality assurance group.

For the twenty-four target project, we applied the rule to classify them. Fourteen projects are then in Good group and ten in Fair. In more precise, four projects are in Average and six projects are in Poor.

3.2 Qualitative results of interview

After the classification of projects in subsection 3.1, we checked common attributes of both Good group and Fair group using the documents of the development plan and report written by an individual project. However, we failed to find common attributes in each group Good and Fair. In order to know the projects' detailed activities and trends that are not described in the reports, we thus hold interviews with software projects managers of both projects of Fair and Good.

The following (a)–(d) were confirmed as the candidates of common attributes that make a project Fair.

(a) A problem concerning quality of the origin (Prior work)

- a-1) A fairly number of problems are found in the origin itself after release.
- a-2) The development activity which aims to improve quality of the origin is not enough.
- a-3) The development activity to check the quality of the origin is not enough.

(b) A problem concerning test and review activity

- b-1) Review and test are done with little consideration to the technical matter specific to communication software.
- b-2) Activity of source code review and a unit test is scarce in some projects.

(c) A problem concerning project management and planning

- c-1) There is much delay before development plan and report are constructed. Moreover, reporting to the quality assurance department is given a rather low priority.
- c-2) The development plan is not detailed enough to be applied actually.
- c-3) The development activity is emphasized and project management tends to become scanty.

(d) A problem concerning specification change

- d-1) Changes of design specification occur even in programming phase and test phases. Software part is sometimes asked for change of interface specification by other related systems.
- d-2) Oversights appear in the operational condition and/or environment, which should be included in the design specification.

4 Statistical Analysis using Logistic Model

4.1 Metrics for Process Quality

We established the detailed evaluation items by which individual project activities are evaluated. The items are selected to express the qualitative attributes discussed in subsection 3.2.

FQ_{total} (the number of problem reports of the product)

- Ft_1: the number of problem reports for twelve months after release

FQ_{org} (the number of problem reports of the origin's product)

- Fo_1: the number of origin's problem reports for twelve months after release per origin's software size

Q_{org} (evaluation of activity for improving the origin's quality)

- O_1: quality of the origin part of software
- O_2: level of examining the origin's quality in design phase
- O_3: level of improving activity for the origin's quality in design and test phases

Q_{rev} (evaluation of review activity)

- R_1: level of consideration to exceptional cases in design review
- R_2: timeliness of design review performed

Q_{test} (evaluation of test activity)

- T_1: level of consideration to exceptional cases and operational conditions in test activity
- T_2: contents of source code review and unit test

Q_{mng} (evaluation of management)

- M_1: adequateness of creating and revising plan
- M_2: adequateness of information included in plan

Q_{spec} (evaluation of specification change)

- S_1: the timing that the specification is fixed
- S_2: preciseness of the specification fixed
- S_3: understandability of requirements and feasibility of design specification

4.2 Logistic Regression Analysis

We evaluated twenty-four projects by the above Q-metrics data. For each project data, we summed up each sub-item in the metric group and got a value of each metric. We then used logistic regression model to analyze the relationships among the metrics and field quality.

The response variable we use to validate the metrics is binary, i.e. was a project's field quality Good or Fair. The value of this response variable is evaluated by the logistic regression model to be described. We took six metrics, FQ_{org}, Q_{spec}, Q_{rev}, Q_{test}, Q_{org} and Q_{mng} as candidates of explanatory variables. For each explanatory variable, the value was obtained by summing up the sub-items' score in each item. Table 1 represents a part of measured metrics data. In Table 1, classification shows the actual result evaluated by the rule in subsection 3.1 using the value of FQ_{total}. As mentioned before, out of the twenty-four projects, fourteen projects are Good and ten projects are Fair.

A multivariate logistic regression model is given in the following formula:

$$E(Y|x_1, \cdots, x_n) = \frac{e^{b_0+b_1x_1+\cdots+b_nx_n}}{1+e^{b_0+b_1x_1+\cdots+b_nx_n}}$$

In this formula, response variable Y represents whether a project is Good or Fair, and explanatory variables x_i are supposed to be selected appropriately. Thus, the value of conditional probability $E(Y|x_1, \cdots, x_n)$ is found.

Table 1. Metrics data

Project	Q_{spec}	Q_{rev}	Q_{test}	Q_{org}	Q_{mng}	Class
1	4	4	5	2	17	Good
2	4	4	5	2	17	Good
3	4	2	5	2	16	Good
4	4	2	6	2	16	Good
5	2	2	5	2	16	Good
6	6	8	15	7	14	Fair
7	6	7	16	7	7	Fair
8	6	4	14	7	15	Fair
9	6	6	14	9	16	Fair
10	4	2	9	6	3	Good
...	...	...	...	...	...	...
24	1	15	24	11	25	Good

Based on the 24 projects' data, we estimated coefficients bi by means of step-wise selection and created a logistic reduction model. As a result, two variables Q_{spec} and Q_{org} were selected as significant.

Deviance was shown to be 19.978 and the degree of freedom was 13. Therefore, the goodness of fit for the model was shown to be good. Moreover, we checked the significance of the model by likelihood ratio test. As a result, it was shown to be somewhat significant $p - value = 0.0095(< 0.01)$. Thus, the model was shown to be significant with significance level 1%.

Using the model for classification and the 24 project data, when $E > 0.5$, the project is classified as Fair and, otherwise, as Good. Twenty projects out of 24 were predicted correctly by the model. If we take into account individual project predicted incorrectly, three projects out of four were Average projects. Although the proposed model did not completely predicted, predictions using our Q_{org} and Q_{spec} metrics appear to be useful for the field quality of communication software.

References

[1] L. J. Arthur, Improving Software Quality - An Insider's Guide to TQM -, John Wiley & Sons, 1993.

[2] R. G. Ebenau and S. H. Strauss, Software Inspection Process, McGraw-Hill, 1993.

[3] J. D. Musa, A. Iannino and K. Okumoto, Software Reliability: Measurement, Prediction, Application, McGraw-Hill, 1987.

[4] O. Mizuno, T. Kikuno, Y. Takagi and K. Sakamoto, "Characterization of Risky Projects based on Project Managers' Evaluation," Accepted to ICSE2000.

[5] V. R. Basili, L. C. Briand and W. L. Melo, "A Validation of Object-Oriented Metrics as Quality Indicators," IEEE Trans. on Software Eng., vol. 22, no.10, pp.751-761, 1996.

[6] R. E. Park, "Software Size Measurement: A Framework for Counting Source Statements", CMU/SEI-92-TR-20, 1992.