Analysis of Work Efficiency and Quality of Software Maintenance Using Cross-Company Dataset*

Masateru TSUNODA†††††, Akito MONDEN††††, Kenichi MATSUMOTO†, Members, Sawako OHIWA†††, and Tomoki OSHINO††††, Nonmembers

SUMMARY Software maintenance is an important activity in the software lifecycle. Software maintenance does not only mean removing faults found after software release. Software needs extensions or modifications of its functions owing to changes in the business environment and software maintenance also refers to them. To help users and service suppliers benchmark work efficiency for software maintenance, and to clarify the relationships between software quality, work efficiency, and unit cost of staff, we used a dataset that includes 134 data points collected by the Economic Research Association in 2012, and analyzed the factors that affected the work efficiency of software maintenance. In the analysis, using a multiple regression model, we clarified the relationships between work efficiency and programming language and productivity factors. To analyze the influence to the quality, relationships of fault ratio was analyzed using correlation coefficients. The programming language and productivity factors affect work efficiency. Higher work efficiency and higher unit cost of staff do not affect the quality of software maintenance.

key words: cross-company dataset, linear regression, work efficiency, working time

1. Introduction

Enterprise software needs software maintenance when a business process is changed. It often occurs, and hence, users sometimes contract software maintenance with companies. Software maintenance does not only mean removing faults found after software release. Software needs extensions or modifications of its functions owing to changes in the business environment, and software maintenance also refers to them. ISO/IEC 14764 [12] classifies software maintenance as follows:

- Corrective maintenance: modifications owing to faults found after software release.
- Preventive maintenance: corrective modifications before potential faults become actual faults after software release.
- Adaptive maintenance: modifications to maintain software availability against environmental changes after software release.
- Perfective maintenance: modifications for conservation or improvement of software performance or maintainability after software release.

A benchmark (reference values to compare an organization’s work efficiency with others [17]) of work efficiency for software maintenance is important. For organizations that offer software maintenance service, benchmarking is the basis of process improvement. Process improvement will enhance price competitiveness of the companies. For users (customers of software maintenance), benchmarking is useful to evaluate the work efficiency of the software-maintenance service supplier. If the work efficiency is low, the price of software maintenance may be higher than that of other service suppliers, and it opens the opportunity to reconsider the contract with the supplier.

In software development, there are many studies which relate to the benchmarking (e.g., [18], [19]), and therefore, software development companies can benchmark their projects. In contrast, on software maintenance, there are few studies which relate to the benchmarking, and therefore, it is difficult to benchmark software maintenance projects. That is, the motivation and the importance of our study is to enable the benchmarking of the maintenance.

In this study, we try to help users and service suppliers benchmark work efficiency for software maintenance. To do that, using a dataset collected from various organizations (a cross-company dataset), those factors affecting work efficiency (e.g., system architecture) are clarified first, and then the dataset is stratified by the factors. On the benchmarking, one compares work efficiency with a reference value whose factor (e.g., system architecture) corresponds to the target. Note that we focused only on software maintenance, and system maintenance is not included. Additionally, we analyzed whether work efficiency affects the quality of software maintenance or not. This is because low quality of the maintenance is problematic, even if work efficiency is high.

Based on a preliminary analysis, we regarded the working time for software maintenance per year as the maintenance cost, and regarded the number of modified modules per year as the amount of maintenance work. Based on
them, we defined work efficiency as the number of modified modules divided by the working time. Although the amount of maintenance work was also measured by the function point (FP) analysis method, it includes many missing values. Therefore, as an alternative, we mainly used the number of modified modules.

One of the major contributions of our study is to illustrate the factors affecting work efficiency based on an analysis that uses a cross-company dataset and the working time recorded on it. Furthermore, we showed the work efficiency and classified the factors. It can be used to benchmark an organization. The other contribution is that we clarified the relationship between work efficiency, cost, and quality of software maintenance. The relationships are not ignorable when considering work efficiency of software maintenance.

We analyzed work efficiency of software maintenance on the study [30]. However, used attributes in the study are different from this study, and the study does not analyze the relationship between quality and work efficiency.

2. Dataset

The dataset used in the analysis includes 134 data points of software maintenance agreements (projects), which were collected from 120 organizations in 2012 by the Economic Research Association. They send questionnaires to companies and, based on the responses, the dataset is made. Hence, we did not know how to record each attribute in detail. Note that, generally, a cross-company dataset is collected in a similar way (e.g., the cross-company dataset [10]). In total, 107 data points are business software and 74 data points are fixed price contracts (software maintenance is performed during certain periods with fixed price [23]). The data points were collected mainly from software-maintenance service suppliers. The number of modified modules and working time were collected in a year.

The attributes analyzed in this study are described in Table 1. In some data points, multiple programming languages were used. We selected the programming language whose usage rate was over 50% of the data points. We assumed that those programming languages whose usage rate was less than 50% did not affect the analysis significantly, because the combination of languages was not very variable. For example, Java and HTML are often used together and, in this case, focusing only on Java is appropriate. The system architecture was settled in the same way. Maintenance type is a classification based on the rate of maintenance activities. When the rate was over 50%, the value of the attribute was set as corrective, preventive, adaptive, or perfective.

In Table 1, the attributes from human factor to tool factor (we call them productivity factors) are defined based on [23], and they were evaluated on a three-point scale (a low value indicates a severe condition, i.e., work efficiency may be decreased). They indicate the degree of difficulty of the factors. The number of analyzed data points was different on each attribute, because each attribute includes missing values. To handle the missing values on regression models, we applied a listwise deletion [16], which is widely used for statistical analysis.

Table 1 Description of Attributes.

| Attribute                          | Description                                                                 |
|------------------------------------|-----------------------------------------------------------------------------|
| Maintenance cost                   | Cost of software maintenance per year (price on each contract)              |
| Working time of service supplier   | Working time for software maintenance of the service supplier per year      |
| Number of engineers of service supplier | Number of engineers of the service supplier (including full-time and part-time workers) |
| Total working time                 | Total working time for software maintenance of the user and the service supplier per year |
| Total number of engineers          | Total number of engineers of the users and the service supplier (including full-time and part-time workers) |
| Total number of modules            | Total number of modules included in the software                           |
| Total FP of software               | Software size measured by FP                                                |
| FP of modified parts               | Amount of modified functions per year measured by the FP analysis method     |
| SLOC of modified parts             | Amount of modified source lines of code per year                            |
| Number of modified modules         | Number of modified modules per year                                         |
| Number of modified screens         | Number of modified screens per year                                         |
| Number of modified reports         | Number of modified reports per year                                         |
| Number of modified data files      | Number of modified data files per year                                      |
| Number of modified batch files     | Number of modified batch files per year                                     |
| Maintenance type                   | Corrective, preventive, adaptive, and perfective                           |
| System architecture                | Mainframe, web system, and client-server                                    |
| Programming language              | SQL, Java, Java Script, Visual Basic, COBOL, HTML, C, JSP, C++, etc.       |
| Business sector                    | Manufacturing, wholesale & retail, banking & insurance, service industry, electronics & computers, etc. |
| Human factor                       | Difficulties about size of project (or organization) and level of skill     |
| Problem factor                     | Difficulties about type, importance, relationships, restriction, and ramification of problems |
| Process factor                     | Difficulties about programming language and software development methodology|
| Productivity factor                | Difficulties about reliability, size, control structures, and complexity of the software |
| Resource factor                    | Difficulties about hardware, duration, and budget                          |
| Tool factor                        | Difficulties about library, compiler, test tool, maintenance tool, and reverse engineering tool |
| Unit cost of staff                 | Unit cost of staff per hour                                                 |
| Number of maintenance cases        | Number of maintenance cases per year                                       |
| Number of faults (fatal, critical, and minor) | Number of faults per year                                                  |
| Fault ratio based on modules       | Number of faults / total number of modules                                 |
| Fault ratio based on FP            | Number of faults / total FP of software                                    |
| Work efficiency                    | Number of modified modules / total working time                            |
| Work efficiency based on FP        | FP of modified parts / total working time                                   |
There are various kinds of software, and this is considered in the analysis. For this, system architecture, programming language, and business sector are used as explanatory variables, and they are considered to denote the kind of software indirectly. For example, when the business sector is banking and the programming language is COBOL, the software is considered as the main software of the banking system.

We defined a new attribute, work efficiency. It is the ratio of output to input of human resources (see Table 1). We treated the number of modified modules as the output and working time as the input. The definition is based on the preliminary analysis described in Sect. 3. As a reference, we also defined work efficiency based on FP of the modified parts.

In Sect. 5, we analyzed the software quality. To analyze it, we defined the fault ratio as the number of faults divided by the number of modified modules. Additionally, we defined the fault ratio based on FP, as a reference (note that the number of cases was small). It is defined as the number of faults divided by FP of the modified parts. The types of faults are fatal fault, critical fault, and minor fault. The faults are not ignorable, because the medians of the numbers of critical and minor faults were not zero.

**Advantage of analyzing cross-company dataset:** The influence of the work efficiency of categorical factors such as programming languages can be classified as follows:

- **Type I:** Although the effect of each category included in a factor (e.g., programming language is a factor, and Java is a category) is different, but the factor affects work efficiency. For example, using Java is the most effective in one company, and using C is the most effective in another company. In such a scenario, although the effects of Java and C are different between the companies, programming language affects work efficiency.
- **Type II:** The effect of each category included in a factor is similar among companies. For example, using Java is the most effective in most companies.

To benchmark maintenance projects as explained in Sect. 7.2 (i.e., estimating work efficiency by focusing on the categories of factors), type II is needed. Therefore, we performed the type II analysis as in previous studies [18], [19]. In addition, the effort estimation model based on the cross-company dataset [20] treats variables as type II. This estimation model is similar to the regression model described in Sect. 4 (i.e., a dependent variable is the working time, and independent variables are programming languages and other factors in both models).

Using the cross-company dataset, we can identify type II factors, and determine the distribution of work efficiency, stratified by categories of the factor (boxplots shown in Sect. 4). If we use the dataset collected from a company to find the relationship between the programming language used and work efficiency, we cannot determine whether the relationship is the same (i.e., type II) in other companies.

### 3. Preliminary Analysis

**Spearman’s rank correlation coefficient:** We used Spearman’s rank correlation coefficient to avoid the influence of outliers. In what follows, “correlation” indicates the Spearman correlation. It is calculated by:

\[ r = 1 - \frac{6 \sum d^2}{n^3 - n} \]  

where \( r \) is correlation, \( n \) is the number of data points, and \( d \) is difference of rank of paired values. To calculate p-value of correlation, the value \( t \) is calculated by:

\[ t = \frac{|r| \sqrt{n-2}}{\sqrt{1-r^2}} \]

The p-value is calculated, assuming the value \( t \) follows Student’s t-distribution with \( n - 2 \) degrees of freedom.

However, the correlation is not suitable to identify non-monotousous relationships. For example, it may be assumed that work efficiency is relatively high when the software size is small, and that when the size is large, work efficiency is also high because more routine work is done. In such cases, the correlation is incorrect.

To consider such non-monotousous relationships, we constructed graphs such as scatter plots in the analysis. With the help of the graphs, relationships can be understood visually. Note that we did not make such graphs for most of the analyses in this section, because they are used for preliminary analysis.

**Multiple linear regression analysis:** We applied the multiple linear regression analysis and analyzed the relationships between independent variables and a dependent variable. It can handle relationships between independent variables (i.e., confounding). In the regression analysis, a model is built using the least squares method. When a dependent variable is denoted as \( y \), and independent variables denoted as \( x_1, x_2, \ldots, \) and \( x_k \), the linear regression model is denoted as:

\[ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \varepsilon \]

where, \( \beta_0 \) is an intercept, \( \beta_1, \beta_2, \ldots, \beta_k \) are partial regression coefficients, and \( \varepsilon \) is an error term.

Using the multiple linear regression analysis has two main advantages. One is that we can compare the explanatory power between models. The other is that we can eliminate confounding between independent variables. In the regression analysis, the ratio scale attributes were log transformed to avoid the influence of outliers. We set the significance level at 0.05.

In the analysis, we did not construct statistical predictive models or apply machine learning methods, which often use many independent variables. However, as explained in Sect. 2, missing values are included in the dataset. Therefore, if we use the many independent variables shown in Table 1, and if the listwise deletion is applied, the number
of available data points becomes smaller, compared with the number of independent variables (the curse of dimensionality may arise). Hence, in the analysis, we gradually identified explanatory variables related to objective variables such as cost. Instead of predictive models, we provide simple benchmarking, as explained in Sect. 7.2.

**Adjusted R²**: As a rule of thumb, when the adjusted R² of the built model is larger than 0.50, the model has adequate explanatory power toward the dependent variable [3]. This threshold is reasonable, especially when analyzing a software project dataset. The project dataset is affected by the human factor to some extent, but this is not a dominant factor. The adjusted R² is calculated by:

\[
R_{adj}^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}
\]  

(4)

where \(R_{adj}^2\) is the adjusted R², \(p\) is the number of independent variables, \(f_i\) is predicted values of the dependent variable, \(y_i\) is actual values of the dependent variable, and \(\bar{y}\) is the average of the dependent variable.

**Partial correlation coefficient**: The p-values of partial correlation coefficients included in the built model is calculated by:

\[
t_i = \frac{\beta_i}{e_i}
\]

(5)

where \(\beta_i\) is partial correlation coefficients, \(e_i\) is standard error (deviation) of \(\beta_i\). The p-values are calculated, assuming the values \(t_i\) follow t-distribution with \(n - 1\) degrees of freedom.

**Multicollinearity**: To check multicollinearity in the model, we used the values of the variance inflation factor (VIF) and condition index. VIF of each independent variable is calculated by:

\[
VIF_i = \frac{1}{1 - R_i^2}
\]

(6)

Where \(R_i^2\) is the coefficient of determination, when we make a regression model for VIF. A dependent variable of the VIF model is \(i\)-th independent variable, and independent variables of the VIF model are the other independent variables.

The condition index of the model is calculated by:

\[
K = \frac{\lambda_{max}}{\lambda_{min}}
\]

(7)

where \(K\) is condition index, and \(\lambda_{max}\) and \(\lambda_{min}\) are the maximum and the minimum eigenvalues of correlation matrix. When VIF of the variables were smaller than 10, and the condition index was smaller than 30, it means there is not multicollinearity in the model [26].

**Path diagram**: To help understanding analysis results, we visually show the overview of the analysis using path diagrams. In the figures, arrows indicate the relationship clarified by the regression analysis, and the numbers by the arrows is correlation coefficients. The overview of the analysis in this section is shown in Fig. 1 to Fig. 3.

### 3.1 Attributes Related to Maintenance Cost

As a preliminary analysis, we analyzed the attributes related to maintenance cost. In the analysis, we focused on the relationship between the working times of engineers, and maintenance cost. For service suppliers, the meaning of working time is almost same as maintenance cost. However, in this study, cost refers to the maintenance cost for users, as shown in Table 1. That is, as shown in Table 1, the cost indicates price. We analyzed the price because users do not know the cost of the service supplier, but only know the price. For users to benchmark maintenance projects (i.e., to validate the price) as explained in Sect. 7.2, analysis of the price is needed.

The price of the services is decided based on the following approaches [22]:

- Cost-based pricing
- Competition-based pricing
- Demand-based pricing

For example, the price is decided based on the value of the service for users in demand-based pricing (i.e., the balance between the money they pay and quality of the service...
they receive). If the price is decided based on demand-based pricing for software maintenance projects, the relationship between the working time and cost (i.e., price) would be weak. Even when the price of software maintenance is decided based on cost-based pricing (i.e., based on the sum of the cost and desired profit), it is not clear whether the desired profit is almost the same among service suppliers. The relationship will be weak if the profit among them is different as well. Therefore, an analysis of the relationship between the working time and maintenance cost is necessary.

If the price is decided based on cost-based pricing, the software maintenance cost for users is mainly based on the labor cost of service suppliers. Thus, it is mainly considered to be settled based on working time or the number of engineers of the service supplier. To observe the effect of the attributes on maintenance cost, we calculated their correlations. Table 2 presents the correlations to maintenance cost.

| Attribute                  | Number of data points | Correlation coefficient | p-value |
|----------------------------|-----------------------|-------------------------|---------|
| Number of engineers of service supplier | 76                    | 0.60                    | 0.00    |
| Working time of service supplier       | 81                    | 0.79                    | 0.00    |

The correlation between working time and number of engineers is 0.56. Thus, they may affect each other. To handle the mutual relationship (i.e., confounding), we applied a multiple linear regression analysis and treated maintenance cost as the dependent variable. Table 3 lists the standardized partial regression coefficients of the model. Working time has a larger coefficient and its p-value is smaller than 0.05. The adjusted R² of the model is 0.73. Therefore, to some extent, it has explanatory power toward maintenance cost.

In the model, VIF were smaller than 10, and the condition index was 8.3. So, there is no multicollinearity in the model. Note that although the number of modified modules affects the working time and number of engineers, it does not affect the maintenance cost directly. Thus, we did not include it as an independent variable.

The results mean that the maintenance cost is mainly settled based on working time of the service supplier. Hence, decreasing working time lessens the maintenance cost. That is, the analyses of working time and work efficiency are regarded as the analysis based on maintenance cost. Although the results are not surprising, it is necessary to enhance the reliability of the analysis. To our knowledge, no study has analyzed the relationships between maintenance cost, working hours, and number of engineers. The results suggest that the cost for users (i.e., the price) is based on cost-based pricing, and multiplying the working time by unit cost is expected to approximate (i.e., the benchmarking explained in Sect. 7.2) the cost.

### 3.2 Attribute Indicating Amount of Modification

To define the work efficiency of software maintenance, an attribute indicating the amount of modification is needed. Although FP of the modified parts is the most appropriate to indicate that amount, it has many missing values. Thus, we identified the attribute that has lesser missing values and has a strong relationship to FP of the modified parts. It is used as the attribute that indicates the amount of modification in the subsequent analyses.

Candidates of the attribute are source lines of code (SLOC) of the modified parts, number of modified modules, number of modified screens, number of modified reports, number of modified data files, and number of modified batch files. The correlations between FP of the modified parts and these candidates are listed in Table 4. The number of modified modules has the strongest correlation and its p-value is smaller than 0.05. Thus, we regarded it as the attribute indicating the amount of modification.

| Attribute                        | Number of data points | Correlation coefficient | p-value |
|----------------------------------|-----------------------|-------------------------|---------|
| Number of modified modules       | 14                    | 0.54                    | 0.04    |
| SLOC of modified parts           | 22                    | 0.37                    | 0.09    |
| Number of modified screens       | 17                    | 0.28                    | 0.28    |
| Number of modified reports       | 13                    | 0.23                    | 0.45    |
| Number of modified data files    | 14                    | 0.46                    | 0.10    |
| Number of modified batch files   | 14                    | 0.06                    | 0.84    |

### 3.3 Relationship between Work Amount and Amount of Modification

We analyzed the relationship between the amount of work and the amount of modification. This is a preliminary analysis of work efficiency. In the analysis, we used the total work amount of the user and service supplier. This is because although maintenance cost is affected by the work amount of the service supplier only, the amount of modification is affected by the total work amount of the user and service supplier. Note that activities of software maintenance are performed not only by the service supplier but also by the
We analyzed the relationship between the number of modified modules and total working time of the user and service supplier. Additionally, we analyzed the relationship between FP of the modified parts and total number of engineers of the user and service supplier. The analysis was performed to validate the use of number of modified modules and total working time to analyze work efficiency.

The correlations between work amount and amount of modification are listed in Table 5 and Table 6. The strength of the relationship between number of modified modules and total working time is moderate. Similarly, the strength of the relationships between other attributes of amount of work and amount of modification is also moderate. That is, the strength of all relationships is almost the same, and it means that the analysis based on number of modified modules and total working time is not inappropriate.

Using a simple linear regression analysis, the relationship between number of modified modules and total working time was analyzed. This is because $R^2$ of the model can be used as the reference value of the subsequent analyses. $R^2$ of the model is 0.32. This means that the number of modified modules is not sufficient to settle the total working time.

### 3.4 Relationship between Work Efficiency and Software Size

When the software size is large, software maintenance may become difficult and it would affect work efficiency. Therefore, we analyzed the relationship between software size (i.e., total number of modules) and work efficiency (i.e., number of modified modules / total working time). In the analysis, it is assumed that the relationship between the software size and work efficiency is monotonous. The assumption is the same as that in COCOMO[3], which is a famous software development effort model. COCOMO assumes that the work efficiency is low when the software size is large (the relationship is called the diseconomies of scale). However, it is uncertain if the assumption on software maintenance is correct. To check the relationship visually, we made a scatter plot of the software size and work efficiency.

The result is presented in Table 7 and Fig. 4. In the figure, values of x-axis was log-transformed, because the distribution is much skewed. As shown in this table and figure, the relation is weak. Additionally, we analyzed the relationship using work efficiency based on FP. It is derived by FP of the modified parts / total working time. Table 8 and Fig. 5 indicate that the relation is not strong. Note that the number of cases was small, and therefore, the result is for reference. Based on this analysis, we did not include software size in the following analysis.

### 4. Analysis of Working Time

In this section, we applied a multiple linear regression analysis. In the models, we treated the total working time of the
user and service supplier as the dependent variable. Using the multiple linear regression analysis, we can consider the confounding factors. That is, the standardized partial regression coefficients are calculated by eliminating the influence of confounding.

In the models, we used the number of modified modules as an independent variable. This means that we can consider the influence of the number of modified modules when building the regression model. Note that we did not take into account the software size (i.e., total number of modules). This is because it did not affect the number of engineers in our previous study [30].

We also showed the relationships between work efficiency and attributes using boxplots. Specifically, based on the standardized partial regression coefficients of the models, those attributes affecting the total working time were identified, and the distributions of work efficiency stratified by the attributes were shown using boxplots. The boxplot is used for the benchmarking.

Additionally, we showed the relationships using the work efficiency based on FP (i.e., FP of the modified parts). When it is similar to the work efficiency based on number of modified models, it supports the results. Note that the number of cases was small, and therefore the result is for reference.

In the boxplots, the bold line in each box indicates the median value. Small circles indicate outliers, that is, values that are more than 1.5 times larger than those in the 25%—75% range from the top of the box edge. Stars indicate extreme outliers, whose values are more than 3.0 times larger than those in this range. Some outliers are not included in the boxplots to improve their readability.

Before applying the multiple linear regression analysis, we made dummy variables to handle the nominal scale attributes. When building the models, we applied stepwise variable selection. The variables were included when the p-value was smaller than 0.05 and excluded when the p-value was larger than 0.1. Due to page limitation, we picked up programming language and productivity factors, which explicitly related to work efficiency. The overview of the analysis in this section is shown in Fig. 6.

### 4.1 Effect of Programming Language

In software development, the programming language affects productivity [19]. It may also affect the work efficiency of software maintenance. Therefore, we focused on the effect of programming language on the total working time. We treated the number of modified modules and programming language as the independent variables, and the total working time as the dependent variable in the multiple linear regression analysis. On the built model, the adjusted R^2 is 0.52. Thus, the use of both the number of modified modules and programming language is effective if users or service suppliers try to settle the total working time through a multiple linear regression model.

Table 9 lists standardized partial regression coefficients of the model. In the variable selection, the dummy variable of the Java language was chosen and other variables of the programming language were eliminated. In the model, the values of the VIF of the variables were smaller than 10, and the condition index was 4.7. This means that there is no multicollinearity in the model. The standardized partial regression coefficients had positive value. Therefore, the total working time is increased when the programming language used on the maintained software is Java.

For benchmarking of software maintenance, the relationship between Java and work efficiency is shown by the boxplot in Fig. 7. In the figure, the distribution of work efficiency depends on whether Java is used or not. Table 10 lists the work efficiency of each programming language. The Java language has the lowest work efficiency. In Fig. 8, we present this relationship using the work efficiency based on FP. Although the median of work efficiency of Java is almost the same as that in other languages, the variance of work efficiency is larger.
efficiency of Java is small, and the distributions shown by the boxplots are similar to those in Fig. 7. Hence, it supports the result.

4.2 Effects of Productivity Factors

We analyzed on the effect of productivity factors on total working time using a multiple linear regression analysis. The factors are similar to the productivity factors in COCOMO [3], and the factors in COCOMO play important roles. Thus, productivity factors may be also important for software maintenance.

On the built model, the adjusted $R^2$ is 0.41. The value is smaller than 0.50, and therefore, the use of number of modified modules and productivity factors is not sufficient to settle the total working time. Table 11 lists the standardized partial regression coefficients of the model. In the model, the values of the VIF of the variables were smaller than 10, and the condition index was 10.0. That is, there is not multicollinearity in the model.

In the variable selection, the tool factor was chosen and other variables of the productivity factors were eliminated. The standardized partial regression coefficients had negative values. Therefore, the total working time is decreased when the value of the tool factor is larger (i.e., the demand for the factor is not severe). When the tool factor (e.g., maintenance tools) can be changed, it should be done to enhance work efficiency.

The relationship between tool factor and work efficiency is shown by the boxplot in Fig. 9. The work efficiency is higher when the values of the factors are 3. Figure 10 shows the work efficiency based on FP. In the figure, work efficiency is lower when the values of the factors are 1, and this tendency is similar to that in Fig. 9. That is, the severity of the tools affects the work efficiency of software maintenance.

4.3 Effects of Multiple Attributes

In Sects. 4.1 and 4.2, we assumed that the users or service suppliers benchmark their activity focusing on a single attribute. To support it, we analyzed the relationships between the total working time and each attribute. For example, to support benchmarking based on programming language, we showed the effect of programming language on total working time using multiple a linear regression analysis. Additionally, we showed the distributions of work efficiency based on the programming language using a boxplot.

In this section, the dominant attributes that affect the total working time are shown. The candidates are attributes that we selected in a preliminary analysis (Note that the analysis is different from the analysis explained in Sect. 3). To perform this, we applied multiple linear regression analysis. Candidates of independent variables are web system (architecture), Java (programming language), tool factor (productivity factors), banking and insurance (business sector),

Table 11 Model using the productivity factors.

| Attribute            | Standardized partial regression coefficients | p-value | VIF |
|----------------------|---------------------------------------------|---------|-----|
| Number of modified modules | 0.44                                       | 0.00    | 1.1 |
| Tool factor          | -0.39                                      | 0.00    | 1.1 |

Table 12 The model using the multiple factors.

| Attribute            | Standardized partial regression coefficients | p-value | VIF |
|----------------------|---------------------------------------------|---------|-----|
| Number of modified modules | 0.47                                       | 0.00    | 1.1 |
| Java                 | -0.37                                      | 0.00    | 1.1 |
| Tool Factor          | -0.29                                      | 0.01    | 1.2 |
adaptive (maintenance type), and the number of modified modules.

Table 12 lists the standardized partial regression coefficients of the model. In the model, the values of the VIF of the variables were smaller than 10, and the condition index was 11.4. Hence, there is not multicollinearity in the model.

The variable selection chose the dummy variable of Java, tool factor, and number of modified modules. It eliminated the dummy variables of the web system, adaptive maintenance, and banking and insurance. Hence, when multiple attributes are considered, they do not affect the total working time. The standardized partial regression coefficients of the tool factor had a negative value. Therefore, the total working time is decreased if the demand for the tool factor is not severe, even when multiple attributes are considered.

On the built model, the adjusted $R^2$ is 0.56. The value is larger than 0.50, and therefore the use of the tool factor, number of modified modules, and dummy variable of Java is sufficient to settle the total working time.

Comparing the absolute value of the regression coefficients, the effect of the number of modified modules on the total working time is the largest, whereas to use or not use Java affects it moderately. The effect of tool factor is the smallest. However, the p-value of the regression coefficient is smaller than 0.05, and therefore, the tool factor also affects the total working time.

5. Analysis of Software Quality

In software maintenance, software quality (i.e., software faults) is also important. Even if the work is efficient and maintenance costs are low, they are meaningless if the number of faults is increased by them. Hence, we analyzed the relationships between software fault ratio, unit cost of staff, and work efficiency. Table 13 lists the average and median of the faults. The overview of the analysis in this section is shown in Fig. 11 to Fig. 14.

5.1 Effect of Work Efficiency

We analyzed the relationship between work efficiency and fault ratio. Both of these metrics are divided by the amount of software modification. As the denominator, we used the number of modified modules and FP of the modified parts. Table 14 lists the relationship based on modules. The relationship of fatal fault ratio was very weak, and the other fault ratio was negatively and weakly correlated with work efficiency.

Table 15 lists the relationships based on FP. As presented in the table, work efficiency was negatively correlated with fault ratio. Figure 15 illustrates the strongest relationship to fault ratio. In the figure, the fault ratio was higher when work efficiency was lower. From the result, when work efficiency is high, at least it does not harmfully affect the fault ratio.

5.2 Attributes Related to Unit Cost

Before analyzing the unit cost of staff, we analyzed other attributes that may be related to unit cost. Both the work efficiency based on number of modules and the efficiency based on FP were used. First, we focused on software size (i.e., total number of modules). The relationship between software size and unit cost is listed in Table 16. The relationship was very weak in both cases. Thus, software size does not affect the unit cost of staff.

Furthermore, we focused on the attributes related to work efficiency. We selected Java and tool factor based on the analysis results explained in Sect. 4.3. The relationships are shown in Fig. 16 and Fig. 17. As shown in the figures, the median of unit cost is almost the same across the categories. Therefore, they do not affect the unit cost of staff significantly.

5.3 Effect of Unit Cost

We analyzed the relationship between unit cost of staff and software quality. Table 17 lists the relationship. In the table, “module based” means that the fault ratio is obtained...
dividing the number of faults by the number of modified modules. Furthermore, “FP based” means that the rate is obtained dividing the number of faults by FP of the modified parts. When we focused on the module-based ratio, the fault ratio was lower, except for the critical fault ratio. When we focused on FP based ratio, the relationship was negative. Especially, the relationship of minor fault ratio was strong. Figure 18 shows the relationship using a scatter plot. Although the results were not consistent between the module-based and FP-based ratios, the results suggest that, at least, the unit cost does not negatively affect the software quality.
Moreover, we analyzed the relationship between unit cost of staff and work efficiency. The strength of the relationship is presented in Table 18. When focusing on work efficiency based on modules (i.e., the numerator is the number of modified modules), there was a negative relationship. Figure 19 illustrates the relationship using a scatter plot. When focusing on the efficiency based on FP (i.e., the numerator is FP of the modified parts), the relationship was weak. Although the relationship was inconsistent, higher unit cost seems to improve work efficiency.

5.4 Effect of Software Size

We analyzed the influence of software size (i.e., total number of modules and total FP of the software). The relationships between the fault ratio and software size are presented in Table 19 and Table 20. When focusing on the fault ratio based on the number of modules, the relationships were weak. When focusing on the rate based on FP, the relationships were not weak, although the number of cases was small. We selected the strongest relationship and visualized it in Fig. 20. The figure also shows that the relationship was not weak. Therefore, software size may affect the fault ratio.

6. Related Work

There are many studies which focused on software maintenance [2], [8]. For example, Sjøberg et al. [25] evaluated software maintenance metrics, applying them to four systems. Wahler et al. [33] showed how refactoring (i.e., software maintenance) process was conducted and evaluated. In contrast, there are very few studies which analyzed work efficiency using cross-company dataset. Some studies have analyzed the work efficiency factors of software maintenance. Jørgensen [15] analyzed the software company dataset and showed that work efficiency is not affected

---

**Table 17** Relationship between unit cost of staff and fault ratio.

| Module based | FP based |
|---------------------------------|---------------------------------|
| Fatal  | Critical | Minor | Fatal  | Critical | Minor |
| Correlation coefficient | 0.09 | 0.27 | 0.10 | -0.28 | -0.04 | -0.62 |
| p-value | 0.55 | 0.08 | 0.51 | 0.37 | 0.90 | 0.03 |
| Number of data points | 43 | 43 | 43 | 12 | 12 | 12 |

**Table 18** Relationship between unit cost of staff and work efficiency.

| | Work efficiency (module based) | Work efficiency (FP based) |
|--------------------------------|--------------------------------|
| Correlation coefficient | -0.30 | 0.06 |
| p-value | 0.05 | 0.85 |
| Number of data points | 42 | 12 |

**Table 19** Relationship between software size and fault ratio based on the number of modules.

| Correlation coefficient | Fatal fault ratio | Critical fault ratio | Minor fault ratio |
|------------------------|-------------------|----------------------|-------------------|
| p-value                | 0.01              | 0.26                 | -0.18             |
| Number of data points  | 47                | 47                   | 47                |

**Table 20** Relationship between software size and fault ratio based on FP.

| Correlation coefficient | Fatal fault ratio | Critical fault ratio | Minor fault ratio |
|------------------------|-------------------|----------------------|-------------------|
| p-value                | 0.18              | -0.66                | -0.66             |
| Number of data points  | 14                | 14                   | 14                |
by the programming language. This result is different from our result. This may be because the work efficiency was measured by the LOC (lines of code) in the study, and it might have affected the result. The author pointed out that the result does not mean that the language is not important for maintenance efficiency, when the influence of LOC on the efficiency is considered. Ahn et al. [1] used variables that are similar to the productivity factors in a software-maintenance effort estimation model. However, these studies did not analyze cross-company datasets.

The International Software Benchmarking Standards Group (ISBSG) collects a cross-company dataset of software maintenance [10]. Tsunoda et al. [32] analyzed the dataset and concluded that only data from some companies can be used to analyze work efficiency on the dataset owing to missing values. In contrast, we analyzed data points collected from many companies (roughly speaking, each data point was collected from each company). Thus, the analysis results of this study are expected to have high generality.

Few reports or studies have analyzed cross-company software maintenance datasets. The Japan Users Association of Information Systems (JUAS) et al. used a cross-company dataset, and showed the work efficiency stratified by business sector [13]. They defined maintenance cases per engineer as work efficiency, and their definition is rather approximate compared with our definition. However, JUAS reports [13], [14] did not analyze the relationships between work efficiency and factors quantitatively.

We analyzed factors related to work efficiency on software maintenance using a cross-company dataset [30]. In the study, work efficiency was defined as the number of modified modules per engineer. The study defined work efficiency based on the number of engineers, and therefore their definitions are rather rough. In contrast, this study defines work efficiency based on working time. Although it is not easy for users to understand working time for software maintenance, the definition is more precise.

There are many studies which focused on quality of software maintenance. For instance, Tarvo [27] proposed a statistical model which estimates the risk of the modification, to avoid regression when Windows operating system is updated. Sneed et al. [24] identified factors which are important for the success of software maintenance. However, as long as we know, there is no study which analyzed the relationships between software maintenance quality and unit cost using cross-company dataset.

Among the studies explained above, studies related to the work efficiency of software maintenance are studies [1], [13], and [15] (i.e., they are strongly related to our study). The followings are the pros and cons of studies [1] and [15]:

- Pros: Software maintenance project effort estimation model is proposed.
- Cons: The studies did not use cross-company datasets, and therefore:
  - The external validity of the models is unclear.
  - The work efficiency values collected from various companies were not incorporated, and the results cannot be used for benchmarking.

The fact that we used a cross-company dataset leads us to believe that our results have an external validity to some extent. In addition, we incorporated the distribution of the work efficiencies collected from various companies.

The followings are the pros and cons of study [13]:

- Pro 1: The study used a cross-company dataset.
- Pro 2: Instead of the number of modified modules and working time, the study used maintenance cases and the number of engineers, which are relatively easy to collect.
- Con 1: The study did not apply multivariate statistical analyses such as multiple regression analysis, and therefore, the validity of the stratification by factors (e.g., the business sector) is not clear.
- Con 2: The definition of the work efficiency (i.e., the maintenance cases per engineer) is approximated, and therefore, estimation of the working time based on the work efficiency is not very accurate.

In contrast, we applied multiple regression analysis to enhance the validity of the stratification (e.g., stratification by Java). In addition, we defined the work efficiency based on the number of modified modules and working time. As a result, estimation of the working time based on our definition was more accurate than that in study [13]. We show a comparison of the accuracies in the Sect. 7.2.

7. Discussion

7.1 Validity of the Findings of the Analysis

In this section, we compare the analysis results to other studies, to evaluate the validity of the findings.

**Influence of programming language**: Past studies [19], [31] showed that programming languages affect the work efficiency in software development. Thus, it is natural that it also affects the work efficiency in software maintenance. In our analysis, using Java lowered the work efficiency. As shown in Table 10, the compared programming languages include lightweight programming language such as JavaScript and SQL, and they are considered to have higher work efficiencies. Moreover, in [31], Visual Basic had higher work efficiency than Java in software development, and the trend would be the same in software maintenance. These may be the reasons why Java had a lower work efficiency.

**Influence of tool factor**: The effort estimation model COCOMO II [4] includes cost drivers that affect the work efficiency of software development. In the drivers, “platform difficulty” signifies constraints of the platform (e.g., how often a DBMS system is updated), and “language and tool experience” indicates for what duration developers use the languages and tools. They are similar to the “tool factor” in our dataset. Therefore, it is probable that such a tool...
factor affects the work efficiency not only in software development, but also in software maintenance.

**Influence of work efficiency and unit cost:** Work efficiency and unit cost do not affect the operation quality (disruption time) of IT operations [29]. Additionally, the unit cost of engineers does not affect work efficiency in software development [30]. Based on these facts, the relationships among work efficiency, unit cost, and software quality shown in our analysis are reasonable. Such relationships might be observed in software lifecycle processes.

**Applicability to other markets:** In the analysis of software development productivity (work efficiency), the factors affecting productivity are similar among countries [17], [31], [34] (e.g., business sector and programming language). Thus, the relationship clarified in this study is expected to be observed in other markets. However, Cusumano et al. [7] showed that productivity is different among countries. Thus, the work efficiency shown in the boxplots should not be regarded as being absolutely true in other markets.

7.2 Applying the Benchmark for Practical Use

Here, we illustrate how to apply the benchmark in practice. If the working time and number of modified modules are known (the case for service suppliers), we can calculate the work efficiency directly, and compare it with the boxplots (i.e., other companies). If the maintenance cost and number of modified modules are known (the case for users), we can calculate the approximate maintenance cost based on the following, and compare it with the actual cost (i.e., the contract price with the service supplier).

1. Count the number of modified modules on the system.
2. Divide the number of modified modules by the work efficiency shown in the boxplots (i.e., Fig. 7 and Fig. 9), to estimate the total working time.
3. Subtract the working time of the user from the estimated total working time of the service supplier.
4. Multiply the estimated working time of the service supplier by the unit cost shown in the boxplots (i.e., Fig. 16 and Fig. 17) to estimate the maintenance cost.

For example, in step 2, if the programming language is Java, the work efficiency is regarded as approximately 0.01. The estimation accuracy can be enhanced by selecting a smaller box. For instance, when the programming language is Java and the tool factor is 3, the work efficiency should be decided based on the programming language, because the size of the box of Java is smaller than that of the tool factor. Although the estimated cost is an approximation, it is useful to validate the maintenance cost.

Note that providing a more formal estimation model would be more appropriate. However, the models built in the analysis did not have sufficient explanatory power, since their $R^2$ values were not very large. Therefore, we did not present the model, in order to avoid abusing the result. Although the boxplots shown in this study do not indicate the estimated cost explicitly, they are useful to show the variance of the estimation. For example, when the tool factor is 2, the box is not very wide, as shown in Fig. 9, and the estimation is somewhat reliable, since the variance of the work efficiency is small. In contrast, the variance of 3 (the value of the tool factor) is large, and hence the estimation using the boxplot is not very reliable.

7.3 Influence of the Definitions of Work Efficiency

**Overview:** As explained in Sect. 7.2, estimating the total working time is required for users to benchmark maintenance projects. However, as explained in Sect. 6, the definition of work efficiency in other studies such as [13] is rather approximate compared with our definition. To check the influence of the definition, we compared the estimation accuracy of the total working time between our study and that in study [13]. We assumed step 2 in Sect. 7.2 (i.e., selecting a smaller box in boxplots), and picked data points where programming language was Java.

**Estimated working time:** Based on our definition of work efficiency, we can estimate the total working time, using the equation

$$t = m/et$$

where $t$ is the estimated total working time, $m$ is the number of modified modules, and $et$ is the median of the work efficiency (i.e., the number of modified modules / total working time).

Based on the definition of work efficiency in study [13], we can estimate the total working time, using the following equation

$$t = 1920 \frac{c}{en}$$

where $c$ is the number of maintenance cases, and $en$ is the median of work efficiency defined in study [13] (i.e., the number of maintenance cases / the number of engineers). In the equation, $mc/et$ is the estimated number of engineers, and 1920 signifies the yearly working hours (i.e., multiplying 8 hours by 20 days and 12 months) per engineer.

**Comparison procedure:** We removed data points where $mn$, $et$, $mc$, or $en$ included missing values (i.e., list-wise deletion). As a result, the number of selected data points was 14. We performed the comparison as follows:

1. Select data points where the used programming language is Java.
2. The median of $et$ is calculated.
3. The median of $en$ is calculated.
4. Using Eq. (8), estimated working time based on our study is calculated.
5. Using Eq. (9), the estimated working time based on study [13] is calculated.
6. The magnitude of relative error ($MRE$) of estimated working time derived in step 4 is calculated.
7. $MRE$ of estimated working time derived in step 5 is
Fig. 21 Procedure of the comparison to study [13]

Table 21 Accuracy of estimated working time based on work efficiency defined by our study and study [13].

|                | Our study | Study [13] |
|----------------|-----------|------------|
| Median MRE     | 0.50      | 4.14       |

8. Median MRE of our study is calculated, using the results of step 6.
9. Median MRE of study [13] is calculated using the results of step 7.

Figure 21 illustrates the above procedure. Note that in step 2 and 3, et and en were calculated by leave-out cross validation (i.e., each estimation target project was removed from the calculation). Steps 4 to 7 were applied to each data point.

Evaluation criterion: As the evaluation criterion, we used the median of MRE [6]. When \( x \) denotes the actual working time and \( \hat{x} \) denotes estimated working time, the criterion is calculated by the following equation:

\[
MRE = \frac{|x - \hat{x}|}{x} \tag{10}
\]

Lower values of the criterion indicate higher estimation accuracies. Following [9], instead of the average of MRE, we used the median, because the average is affected by outliers [21].

Result: The median of the MRE of each method is shown in Table 21. As shown in the table, the median of our method is smaller than that in study [13]. Therefore, the definition of work efficiency based on the working time is more accurate in estimating the total working time, compared with the definition in study [13].

8. Conclusions

In this study, we analyzed the relationships between work efficiency and attributes. Furthermore, we analyzed the quality of software maintenance. Major findings of the analysis are the following:

- Higher work efficiency does not negatively affect software quality.
- Higher unit cost of staff does not negatively affect software quality, but does not enhance work efficiency.

The results of the analysis are useful for users and service suppliers in order to benchmark their activities with the boxplots shown in this study. Note that the benchmarking should be used as reference, but not as rigid criteria, as the variance of work efficiency is not small.

Acknowledgments

This research was partially supported by the Japan Society for the Promotion of Science (JSPS) [Grants-in-Aid for Scientific Research (A) (No.17H00731)].

References

[1] Y. Ahn, J. Suh, S. Kim, and H. Kim, “The software maintenance project effort estimation model based on function points,” Journal of Software Maintenance: Research and Practice, vol.15, no.2, pp.71–85, 2003.
[2] R. Banker, S. Datar, and C. Kemerer, “A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects,” Management Science, vol.37, no.1, pp.1–18, 1991.
[3] B. Boehm, Software Engineering Economics, Prentice Hall, 1981.
[4] B. Boehm, C. Abts, A. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, Software Cost Estimation with Cocomo II, Prentice Hall, 2000.
[5] R. Brook and G. Arnold, Applied Regression Analysis and Experimental Design, CRC Press, 1985.
[6] S. Conte, H. Dunsmore, and V. Shen, Software Engineering, Metrics and Models, Benjamin/Cummings, 1986.
[7] M. Cusumano, A. MacCormack, C. Kemerer, and B. Crandall, “Software development worldwide: the state of the practice,” IEEE Softw., vol.20, no.6, pp.28–34, 2003.
[8] S. Delughani and N. Hajrahimi, “Which Factors Affect Software Projects Maintenance Cost More?” Acta Informatica Medica, vol.21, no.1, pp.63–66, 2013.
[9] K. Dejaeger, W. Verbeke, D. Martens, and, B. Baesens, “Data Mining Techniques for Software Effort Estimation: A Comparative Study,” IEEE Trans. Softw. Eng., vol.38, no.2, pp.375–397, 2012.
[10] International Software Benchmarking Standards Group, https://www.isbsg.org/
[11] International Software Benchmarking Standards Group (ISBSG), ISBSG Maintenance & Support Data Suite Release 4, ISBSG, 2010.
[12] ISO/IEC 14764:2006, Software Engineering - Software Life Cycle Processes – Maintenance, International Organization for Standardization, 2006.
[13] Japan Users Association of Information Systems (JUAS), Software
Metric Survey, JUAS, 2008 (in Japanese).
[14] Japan Users Association of Information Systems (JUAS), Software Metric Survey 2013, JUAS, 2013 (in Japanese).
[15] M. Jørgensen, “Experience With the Accuracy of Software Maintenance Task Effort Prediction Models,” IEEE Trans. Softw. Eng., vol.21, no.8, pp.674–681, 1995.
[16] R. Little and D. Rubin, Statistical Analysis with Missing Data (2nd ed.), John Wiley & Sons, 2002.
[17] C. Loken, T. Wright, P. Hill, and M. Stringer, “Organizational Benchmarking Using the ISBSG Data Repository,” IEEE Softw., vol.18, no.5, pp.26–32, 2001.
[18] K. Maxwell and P. Forselius, “Benchmarking software development productivity,” IEEE Softw., vol.17, no.1, pp.80–88, 2000.
[19] K. Maxwell, L. Wassenhove, and S. Dutta, “Software Development Productivity of European Space, Military, and Industrial Applications,” IEEE Trans. Softw. Eng., vol.22, no.10, pp.706–718, 1996.
[20] E. Mendes and C. Loken, “Replicating studies on cross- vs single-company effort models using the ISBSG Database,” Empirical Software Engineering, vol.13, pp.3–37, 2008.
[21] D. Port and M. Korte, “Comparative studies of the model evaluation criteria mmre and pred in software cost estimation research,” Proc. international symposium on Empirical software engineering and measurement (ESEM), pp.51–60, 2008.
[22] K. Rao, Services Marketing, Pearson, 2011.
[23] Software Evolution Research Consortium (SERC), Software Evolution Research Consortium Report, SERC, 1995.
[24] H. Sned and P. Brossler, “Critical success factors in software maintenance: a case study,” Proc. International Conference on Software Maintenance (ICSM), pp.190–198, 2003.
[25] D. Sjøberg, B. Anda, and A. Mockus, “Questioning software maintenance metrics: a comparative case study,” Proc. international symposium on Empirical software engineering and measurement (ESEM), pp.107–110, 2012.
[26] B. Tabachnick and L. Fidell, Using Multivariate Statistics (3rd ed.), Harper Collins College Publishers, 1996.
[27] A. Tarvo, “Mining Software History to Improve Software Maintenance Quality: A Case Study,” IEEE Softw., vol.26, no.1, pp.34–40, 2009.
[28] M. Tsunoda, A. Monden, K. Matsumoto, S. Ohiwa, and T. Oshino, “Analysis of Attributes Relating to Custom Software Price,” Proc. International Workshop on Empirical Software Engineering in Practice (IWSEP), pp.16–22, 2012.
[29] M. Tsunoda, A. Monden, K. Matsumoto, S. Ohiwa, and T. Oshino, “Benchmarking It Operations Cost Based on Working Time and Unit Cost,” Science of Computer Programming, vol.135, pp.57–87, 2017.
[30] M. Tsunoda, A. Monden, K. Matsumoto, and T. Oshino, “Analysis of Software Maintenance Efficiency Focused on Process Standardization,” Proc. International Workshop on Empirical Software Engineering in Practice (IWSEP), pp.3–8, 2011.
[31] M. Tsunoda, A. Monden, H. Yadohisa, N. Kikuchi, and K. Matsumoto, “Software Development Productivity of Japanese Enterprise Applications,” Information Technology and Management, vol.10, no.4, pp.193–205, 2009.
[32] M. Tsunoda and K. Ono, “Pitfalls of Analyzing a Cross-company Dataset of Software Maintenance and Support,” Proc. International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp.349–354, 2014.
[33] M. Wahler, U. Drofenik, and W. Snipes, “Improving Code Maintenability: A Case Study on the Impact of Refactoring,” Proc. International Conference on Software Maintenance and Evolution (ICSM), pp.493–501, 2016.
[34] H. Wang, H. Wang, and H. Zhang, “Software Productivity Analysis with CSBSG Data Set,” International Conference on Computer Science and Software Engineering, pp.587–593, 2008.

Masateru Tsunoda  is an associate professor in the Department of Informatics at Kindai University, Japan. His research interests include software measurement and human factors in software development. Tsunoda received a Doctor of Engineering in information science from the Nara Institute of Science and Technology. He is a member of IEEE, the Institute of Electronics, Information, and Communication Engineers, the Information Processing Society of Japan, JSSST; and the Japan Society for Information and Systems in Education.

Akito Monden  is a professor in the Graduate School of Natural Science and Technology at Okayama University, Japan. His research interests include software measurement and analytics, and software security and protection. Monden received a Doctor of Engineering in information science from Nara Institute of Science and Technology. He is a member of IEEE, the Institute of Electronics, Information, and Communication Engineers, the Information Processing Society of Japan, and JSSST.

Kenichi Matsumoto  is a professor in the Graduate School of Information Science at Nara Institute of Science and Technology, Japan. His research interests include software measurement and software process. Matsumoto received a PhD in information and computer sciences from Osaka University, Japan. He is a fellow of the Institute of Electronics, Information, and Communication Engineers and the Information Processing Society of Japan, a senior member of IEEE, and a member of ACM and JSSST.

Sawako Ohiwa  received the B.E. degree (1991) in literature from Kyushu University. She joined Economic Research Association (ERA) in 1991. She works at Economic Research Institute, ERA from 2010. She is a general manager of the second investigation laboratory from 2018. Her research topics are collecting and analyzing data of software development and maintenance projects. Especially, the productivity and the cost of the projects is her major research interests. She is a member of SPM, JFPUG, SERC, and JASA. She is a certified function point specialist.

Tomoki Oshino  received the B.E. degree (1981) in engineering from Kitami Institute of Technology. He joined Japan National Railway in 1981, and joined Economic Research Association (ERA) in 1986. He works at Economic Research Institute, ERA from 2008. His research topics are collecting and analyzing data of software development and maintenance projects. Especially, the productivity and the cost of the projects is his major research interests. He is a member of IPSI, SPM, JFPUG, SERC, and...