Software development productivity of Japanese enterprise applications

Masateru Tsunoda
Graduate School of Information Science,
Nara Institute of Science and Technology,
8916-5 Takayama, Ikoma, Nara, Japan

Akito Monden
Graduate School of Information Science,
Nara Institute of Science and Technology,
8916-5 Takayama, Ikoma, Nara, Japan

Hiroshi Yadohisa
Faculty of Culture and Information Science,
Doshisha University,
1-3 Miyakodani, Tatara, Kyotanabe, Kyoto, Japan

Nahomi Kikuchi
Software Engineering Center, Information-technology Promotion Agency,
2-28-8 Hon-Komagome, Bunkyo-ku, Tokyo, Japan

Ken-ichi Matsumoto
Graduate School of Information Science,
Nara Institute of Science and Technology,
8916-5 Takayama, Ikoma, Nara, Japan

http://www.springerlink.com/content/64063k1t62442460/

Abstract To clarify the relationship between software development productivity and the attributes of a software project, such as business area, programming language and team size, this paper analyzed 211 enterprise application development projects in Japan using a software engineering data repository established by the Software Engineering Center (SEC), Information-technology Promotion Agency, Japan. In the analysis, we first identified factors that related to productivity based on a parallel coordinate plot (PCP) and a one-way ANOVA. An in-depth analysis on each productivity factor was then conducted by selecting a project subset for each factor so that the effect of other factors is minimized. Our findings include that the average team size was the strongest attribute relating to productivity. The outsourcing ratio (percentage), which can be controlled by software development companies, and the business sector both showed a moderate relationship to productivity. Finally, product size (FP), the duration of development and the programming language were only weakly related to productivity.

Keywords Software productivity analysis – Enterprise software – ANOVA – Variance explained – Outsourcing ratio

1. Introduction

Software development productivity is the fundamental determinant of the profitability of software vendors as these companies need to meet today's growing demand for large scale but short life-cycle software applications. However, various attributes of software projects influence productivity, such as business area, application type, programming language, team size and experience, etc. [3][11][12][15]. Therefore, before these companies can increase their productivity, they need to be aware of the dominant attributes that influence productivity and to be certain which of these are controllable and which are not.

From the perspective of software project benchmarking, including productivity baseline analysis, the Software Engineering Center (SEC), Information-technology Promotion Agency, Japan, has recently started to build a software engineering data repository (called the SEC repository in this paper) consisting of information on completed Japanese software development projects. This repository has the financial support of the Ministry of Economy, Trade and Industry (METI) in Japan [17] and contains information collected from 1009 projects and 15 software vendors in 2004-2005.

The goal of this paper is to clarify the relationship between the various attributes of software projects and productivity in Japan. Using the SEC repository, this paper extends a previous analysis reported in a SEC white paper [16]; it also adds to results reported in our former short paper [18] by using statistical techniques (e.g. ANOVA and variance explained) to conduct further analyses of attributes relating to software productivity. We selected 211 of 1009 new software development projects (excluding maintenance projects and re-development projects) in the SEC repository of projects for enterprise "application software" development (excluding embedded software, consumer software, middleware software, operating systems, etc). In the analysis, we first transform the ratio scale attributes into nominal scale attributes to ensure all of the attributes have equal weight on a nominal scale. This transformation enables us to apply

a one-way ANOVA to each of the attributes in order to clarify how variance in productivity can be explained by each attribute. However, ANOVA is insufficient for multivariate data analysis because it does not consider the complicated relationships among explanatory variables. Therefore, we perform further analysis to each explanatory variable by selecting a project subset for each variable so that the effect of other variables on the objective variable (i.e. productivity) is minimized.

So far, software productivity analyses have been performed in many different countries [3][11][12][15]. For Japanese companies, several empirical analyses have been made [3][6]; however, these studies focused on clarifying the differences in software development types (e.g. platform, customer, software type, etc.) between Japan and other companies rather than focusing on the productivity attributes. Also, the number of projects analyzed was relatively small (Blackburn et al. used 49 projects of the United States and Japan [3], and Cusumano et al. did 27 projects of Japan [6]). It was because Japanese companies did not want to put their productivity data into public. However, now the situation has changed. A Japanese national project to collect software project data has started in 2004; and fortunately, we were able to analyze its dataset (SEC repository) in this paper.

Since the software development environment and culture are different in each country, we might have different results from different datasets. One of the distinctive properties of Japanese enterprise software development is that many projects have high outsourcing ratio (percentage). As far as we know, there was no past research that focused on the relationship between outsourcing ratio and productivity. In addition, because our project dataset is relatively new (most projects were completed in 2004), there are “intranet/web” architecture projects in our dataset. This architecture type is quite new, and thus not included in past researches. So the key contributions of our research is analyzing the relationship between outsourcing ratio and productivity as well as software architecture and productivity, with the

comparison of Japanese dataset and other countries' ones.

From another point of view, in empirical software engineering research, a "replicated study" with different data set is considered extremely important [1][2][13]. It is because, there are so many unclear factors in software development, e.g. human factors and business factors, so that an analysis result from only one data set is not trustworthy enough to reflect the typical software development in the world. In our case, we believe that productivity analysis in Japan is needed as well as that in the United States or in other countries, to clarify both common productivity factors in all datasets and specific factors in Japan.

In Section 2, we introduce the SEC repository. Next, research theory is explained in Section 3. Then, in Section 4, we analyze the relationships between each project attribute and productivity. In Section 5, we describe further analyses taking attribute correlations into account. We discuss analysis results in section 6. Finally, conclusions and future research topics are outlined in Section 7.

2. The SEC repository

2.1. Overview

The 2005 version of the SEC repository consists of 1009 projects from 15 Japanese software companies [16]. Project completion dates ranged from 1996 to 2005 (most were completed in 2004). All of the projects are custom enterprise software developments, the type of development project dominant in Japan [6][7]. Each project is characterized by about 100 major attributes (metrics) and 300 related attributes, although many include a large number of missing values.

The SEC repository has a strict data quality assurance program. Software companies who provided their project data to the SEC repository must join the data analysis meeting several times a year to clarify and improve the data definition and the way of data collection. After data

were collected, each attribute of a project in the SEC repository was inspected by SEC members to make sure recorded values are not erroneous. If the SEC members found a suspicious value (that can be considered an outlier) in the repository, they confirmed the data provider whether it is a true value or there is some mistake. By such a data quality assurance program, the SEC dataset is continuously updated, and thus we believe it reasonably good as an analysis target.

2.2. Analysis target

We selected projects for analysis based on the following criteria (using these criteria is recommended for analysis in the SEC's white paper [16]).

- Projects that had effort values (i.e. non-missing values) in all five basic phases: architectural design, detailed design, coding, integration testing, and system testing. Sums of effort values of these five phases were used in the productivity analysis.
- Projects whose function points (FP) were recorded as a project size attribute. (Note that FP is a necessary attribute for calculating productivity.)
- Projects using IFPUG (or a similar method) as a FP calculation method. This is required to ensure that selected projects have the same measures of product size.
- New development projects. (Maintenance projects and enhancement projects were excluded.)

As a result, 211 projects were selected for this analysis. The type of software developed in these projects was "application software." (No consumer software, middleware software or operating system software development projects were included.) The waterfall development process was used in all but 2 of the projects.

In this paper, "productivity" is defined as FP divided by (total) development effort spent for

the five basic phases (person-hour). Intuitively, productivity is the amount of functionality a person can contribute in an hour. Table 1 shows the productivity statistics for the 211 projects; Fig. 1 shows the distribution of these statistics. The productivity of the upper quartile projects is about 3.3 times higher than that of the lower quartile projects. Because of a skewed productivity distribution, this paper uses non-parametric tests in the analysis.

Here we compare the average productivity of our SEC dataset with other countries' datasets. Jones [8] showed that the average productivity of 1065 MIS projects (including 505 new projects and 560 enhancement projects) in the United States (from 1995 to 1999) was 0.07 FP/staff-hours. Premraj et al. [15] showed that the productivity of 401 projects completed between 1997 and 2003 in Finland was 0.23 FP/staff-hours on average. As shown in Table 1, the average productivity was 0.15 FP/staff-hours in our SEC repository. We must be careful to give any interpretation to this result because phases in which effort values are recorded are not exactly the same between these datasets, and also each dataset contains different types of systems.

Table 2 shows the project attributes used in our analysis. Most of these attributes are not under the control of software vendors, except for the average team size and the outsourcing ratio. Here, the outsourcing ratio is defined as effort outsourced to subcontracting companies divided by the total development effort. The effort for an outsourced part was recorded based on a report from a subcontracting company who was responsible for that part. However, in case the effort for an outsourced part was unknown or uncertain, it was estimated from billing. The lower quartile for the outsourcing ratio is 0.6, which is quite high. Such high outsourcing ratio is one of the distinctive properties of Japanese enterprise software development.

2.3. Characteristics of high and low productivity projects

Before conducting statistical analyses, we used a parallel coordinate plot (PCP) [9], which is a common technique for representing high-dimensional data, to visually determine the major

differences between high and low productivity projects. To visualize a set of projects in an n-dimensional (i.e. n-attribute) space, a PCP describes n parallel vertical line segments, where the n-th line corresponds to the n-th project attribute; a project is represented as a polyline with vertices on the parallel line segments. The top of a vertical line signifies the maximum value of an attribute and the bottom signifies the minimum value.

Fig. 2[1] is a PCP for high productivity projects. Note that ratio scale attributes were log transformed and that missing values were assigned as minus values (the bottom section of the vertical lines) in the PCP. From this figure, we can see that many high productivity projects seem to have a small average team size, operate in the "manufacturing" business sector, and create stand alone and 2-layer C/S architectures. Also, a few of these projects have low outsourcing ratios.

Fig. 3 is a PCP for low productivity projects. We can see that many of these projects have large FPs, are of long duration, have a large average team size, are in the "finance/insurance" business sector, and develop intranet/web architectures.

The major distinctions between high and low productivity projects in PCPs are the average team size, architecture, business sector, and outsourcing ratio.

3. Research theory

3.1. Principles for Data Analysis

We believe that any analysis method should be well-matched to the nature of the target dataset that needs to be analyzed. In our case, the most important characteristics of the dataset (software project repository) is that it contains many missing values. It is quite common that the software project repository contains a lot of missing values because it is recorded by hand

[1] These PCPs were made using DAVIS [8].

and gathered from various software project held in different divisions. Our repository also has many missing values (e.g. the missing value ratio is 39% in programming language as shown Table 2).

To analyze such a dataset containing a lot of missing values, model-based analysis methods are not applicable. For example, a multivariate regression model (OLS or other variants), which are often used in other research areas, could not be used for a software project repository because they assume a dataset having no missing values. Therefore, we used one-way ANOVA, which can be applied to a dataset including missing values. For this reason, past studies on software project data analysis also used ANOVA [11][15].

However, ANOVA is insufficient for multivariate data analysis because it does not consider the complicated relationships among explanatory variables. Therefore we performed further analysis to each explanatory variable by selecting a project subset for each variable so that the effect of other variables on the objective variable (i.e. productivity) is minimized. Analysis procedure is described in the following sections.

3.2. Analysis based on ANOVA and variance explained

We identify attributes relating to productivity by the following steps:

Step 1. Transforming ratio scale attributes into nominal scale attributes, each of which had three categories (“low”, “medium” and “high”). This ensured that all of the attributes became nominal scale variables. This transformation enables balanced evaluation of how productivity variance can be explained by each attribute (based on ANOVA.)

Step 2. Applying a one-way ANOVA to each attribute (and to productivity) and calculating the *p*-value and the adjusted variance explained. Then, identifying attributes that have a significant relationship with productivity.

Step 3. Drawing boxplots for attributes that had significant relationships in Step 2 to visualize the relationships between the attribute categories and productivity. A Mann-Whitney U

test [4] (at the 0.05 level) was used to statistically evaluate the differences between the categories.

In Step 1, for each ratio scale attribute, projects whose attribute values were equal to or less than the lower quartile were classified as being in the "low (short, or small)" category on a nominal scale. Similarly, attribute values equal to or greater than the upper quartile were classified as being in the "high (long, or large)" category; the remaining values were classified as being in the "medium" category. The thresholds for these categories are shown in Table 3.

In Step 2, if a *p*-value from an ANOVA is significant, it means there is a relationship between an attribute and productivity. In this paper, the significance level is set at 0.05. Adjusted variance explained (ω^2), on the other hand, indicates the *strength* (magnitude) of the relationship. It is calculated using the following equation [19].

$$\omega^2 = \frac{SSB - (k-1)MSE}{SST + MSE} \qquad (1)$$

In the equation, *SSB* is the sum of squares between groups, *SST* is the sum of squares total, *MSE* is the mean square error, and k is the number of groups. A larger ω^2 indicates a stronger relationship between an attribute and productivity.

3.3. Analysis considering correlated attributes

Based on the analysis in Section 3.2, candidates of productivity factors will be identified. However, since there are correlations among factors, e.g. most large FP projects may have long project duration and vice versa, we need to take such correlations into account to clarify which factors are significant ones. Therefore, next we conduct further analysis on each project attributes by minimizing the effect of other attributes.

In this analysis, for each explanatory variable (e.g. programming language), we build a

project subset by removing projects that possess an attribute that has strong correlation with the productivity (e.g. removing "finance/insurance" business sector projects). By using such a project subset, we then analyze the relationship between an explanatory variable (programming language) and the objective variable (productivity) with minimum effect of other explanatory variable.

Note that there may be an alternative way to remove the effect of unfocused factors by selecting a project subset having identical attributes (e.g. selecting projects of "large FP" and "Java"); however, this approach is not applicable because the number of projects in the subset becomes too small in this case.

4. Analysis results of ANOVA

The results of ANOVA with variance explained are shown in Table 4. Letters in italics indicate p-values < 0.05. Seven attributes (average team size, architecture, business sector, outsourcing ratio, project duration, programming language, and FP) were related to productivity, while two attributes (system user and use of ERP package) were not. The most related attribute was the average team size (ω^2=0.39), while product size (FP) showed a much lower relationship (ω^2=0.05). This result confirms the previous observation from the PCPs that the major distinctions between high and low productivity projects are average team size, architecture, business sector and outsourcing ratio.

Interestingly, although it is intuitive to think that a larger product is more difficult to develop (i.e. the productivity is low), this result indicates that product size is not a major factor related to productivity. Instead, team size has a much greater relationship to productivity and could be considered as a measure of the communication overhead between team members.

4.1. Average team size

Fig. 4 shows boxplots of productivity for three project groups (average team size=small, medium and large). 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 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 this range.

Obviously, as the average team size increases, productivity worsens. The median value for productivity in the low group was about 4.8 times larger than that in the high group. Based on the Mann-Whitney U test, we confirmed that the differences among these groups were statistically significant at the 0.05 level. This result suggests that it is difficult to achieve high productivity with a large development team (more than 10 team members in this case).

4.2. Architecture

Fig. 5 shows boxplots for five different architectures. The stand alone group showed the highest productivity; the mainframe group and the intranet/web (web application) group showed the lowest productivity. The median productivity value of the stand alone group was about 4.1 times larger than that of the mainframe and the intranet/web groups. The differences in productivity between these five groups were statistically significant, except for the difference between the mainframe and intranet/web groups and between the 2-layer C/S and 3-layer C/S groups.

4.3. Business sector

Fig. 6 shows boxplots for five different business sectors. Projects in the manufacturing group had the highest productivity and those in the finance/insurance group had the lowest productivity. The median level of productivity in the manufacturing group was about 2.4 times

larger than that of the finance/insurance group. Statistically significant differences in productivity level were seen between projects in the manufacturing and wholesale/retail groups, between the manufacturing and finance/insurance groups, between the manufacturing and service groups, between the finance/insurance and communications groups, and between the finance/insurance and service groups.

4.4. Outsourcing ratio

Fig. 7 shows boxplots for projects with three different levels of outsourcing ratios. Projects with a low outsourcing ratio (under 0.60) showed relatively higher productivity than projects that had medium and higher outsourcing ratios. The median productivity level in the group with the lowest outsourcing ratio was about 2.0 times larger than that of the group with the highest outsourcing ratio (the difference is statistically significant).

4.5. Programming language

Fig. 8 shows boxplots for four major programming languages. Obviously, the COBOL group had the lowest productivity. The median level of productivity in the Visual Basic group was about 2.3 times larger than that of the COBOL group (the difference is statistically significant).

4.6. Project duration

Fig. 9 shows boxplots for projects with three different durations. As shown in the figure, the longer the development project continues, the lower the productivity. The median level of productivity for the short duration group was about 2.1 times larger than that for the long duration group (the difference is statistically significant).

4.7. Function point

Fig. 10 shows boxplots for three product size (FP) groups. The median level of productivity of the small FP group was about 1.4 times larger than that for the large FP group. The productivity of the large FP group was significantly different from that of the other two groups.

5. Analysis results considering correlated attributes

Based on the analysis results in Section 4, we built project subsets each excluding different project group that had a strong relationship to productivity. For example, one subset can be built by excluding finance/insurance projects, which were strongly related to low productivity. By using this subset, relationship between productivity and other attribute (e.g. the average team size) is more precisely analyzed with the minimum effect of finance/insurance projects. We built a series of such subsets by excluding projects of either “manufacturing” business sector, “finance/insurance” business sector, COBOL language, Visual Basic language, stand alone architecture, mainframe architecture, intranet/web architecture, high outsourcing ratio, low outsourcing ratio, large FP, small FP, long duration, short duration, large average team size, or small average team size.

Fig. 11,…,Fig. 17 shows analysis results for each project attributes. In these Figures, the x-axis shows excluded project categories, and the y-axis indicates productivity, and each line denotes the median of productivity in each project subset. In case the median of productivity greatly changed (i.e. a line in the Figure intersected other lines) when one category was excluded, this means a project attribute being analyzed is related to the excluded category, and thus, further analysis is required.

Below describes our detailed analysis results.

5.1. Average team size

Fig. 11 shows productivity of projects classified by average team size. The productivity lines were not intersected each other regardless of a project category excluded. Therefore, we confirmed that smaller average team size projects were related to higher productivity, as mentioned in section 4.1.

5.2. Architecture

Fig. 12 shows productivity of projects classified by architecture. Regardless of a category excluded (i.e., in all project subsets in x-axis), productivity of stand alone architecture projects was highest, 2-layer C/S was second-highest, and 3-layer C/S was third-highest (although the difference between 2-layer C/S and 3-layer C/S was small). Notably, when "average team size - small" projects were excluded, the difference between stand alone and C/S became small. This indicates that most of stand alone projects had high productivity because of small average team size. But still, stand alone projects are considered high productive. From these result, we confirmed that productivity of stand alone projects was highest, C/S projects was second-highest, and others (mainframe and intranet/web) was the lowest.

5.3. Business sector

Fig. 13 shows productivity of projects classified by business sector. Except for wholesale/retail projects, in all project subsets in x-axis, productivity of manufacture projects was highest, communications projects was second-highest, service projects was third-highest, and finance/insurance business sector projects was lowest. On the other hand, productivity of wholesale/retail projects greatly changed (became very high) when intranet/web projects were excluded. We found that 10 out of 16 wholesale/retail projects were intranet/web architecture, which was low productive. For the further analysis, we selected intranet/web projects only, and

then, compared productivity among different business sectors (Fig. 18). From Fig. 18 and Fig. 13, wholesale/retail projects were considered not low productive in intranet/web architecture projects as well as in other architecture projects. From these results, we could conclude that productivity of manufacturing projects was high, finance/insurance projects was low, and others was medium.

5.4. Outsourcing ratio

Fig. 14 shows productivity of projects classified by outsourcing ratio. Obviously, there is no significant difference between “high” and “medium” outsourcing ratio projects.

On the other hand, “low” outsourcing ratio projects showed higher productivity than “medium” and “high” projects in most cases. Just one case where “low” and “high” outsourcing ratio projects showed almost same productivity was that the case “FP - large” was excluded. This indicates, for small projects, outsourcing ratio did not affect the productivity. For the further analysis, we selected “FP - large” projects only in Fig. 19. As shown in Fig. 19, “low” outsourcing ratio projects showed very high productivity while “high” outsourcing ratio projects showed low productivity. This indicates, for large projects, outsourcing ratio greatly affects the productivity. From these result, it can be considered that the management overhead becomes serious when a large project is outsourced.

5.5. Programming language

Fig. 15 shows productivity of projects classified by programming language (COBOL, Visual Basic, Java and C). As shown in the Figure, COBOL projects showed lowest productivity regardless of a category excluded in x-axis. Hence, COBOL is considered the lowest productivity among four languages.

Interestingly, productivity of Java projects changed (became high) when intranet/web projects were excluded. We found that 12 out of 21 Java projects were intranet/web architecture, which was low productive. For the further analysis, we selected intranet/web projects only, and then, compared the productivity among four languages (Fig. 20). From Fig. 20 and Fig. 15, Java projects were considered high productive in intranet/web architecture projects as well as in other architecture projects.

Another interesting findings is that productivity of C projects became as low as COBOL when “FP - small” projects or “Average team size - small” projects were excluded. This indicates, large C projects are low productive while small C projects are not.

From these results, we could conclude that productivity of Visual Basic and Java projects was high, COBOL projects was low, and C depends on the project size.

5.6. Project duration

Fig. 16 shows productivity of projects classified by project duration. Productivity of “long” duration projects was the lowest regardless of a category excluded in x-axis. Therefore, we could conclude that long duration projects were low productive.

5.7. Function point

Fig. 17 shows productivity of projects classified by FP. Productivity of small FP projects was highest in most project subsets. However, when small FP projects were developed with large team size (i.e. “average team size - small” was excluded in Fig. 17) or with long duration (i.e. “duration - long” was excluded in Fig. 17), then the productivity became as low as or lower than large FP projects. It seems natural that productivity becomes worse when small FP project required large team size. Such projects might be whether unsuccessful projects or high reliable system projects, but unfortunately, we could not conduct further analysis since we did not have

additional information for these projects.

Similarly, while large FP project showed low productivity in most cases, they became high productive when they required only small development team (in Fig. 17, “average team size - large” was excluded) or they required only short duration (in Fig. 17, “duration - long” was excluded). Unfortunately, we could not conduct further analysis due to lack of additional information. We could conclude that larger FP relates to lower productivity with some exceptions.

6. Discussions

6.1. Contributions of research

Software productivity studies have been conducted in many different countries [3][5][6][11][12][15]. The key contributions of our research is analyzing the relationship between outsourcing ratio and productivity as well as software architecture and productivity, with the comparison of Japanese dataset and other countries’ ones.

Our result revealed that the architecture had strong relationship to productivity, whereas Maxwell et al. [12] and Premraj et al. [15] showed that architecture had a relatively weak relationship to productivity (the variance explained was 0.13 [12] and 0.06 [15]). In our result, “intranet/web” architecture projects had low productivity. This architecture type is quite new, and thus, not included in past researches. One interpretation is that intranet/web systems are often developed to replace old infrastructure systems, which used to be developed on mainframe computers in the past. Such replacement might cause low productivity.

While high outsourcing ratio is one of the distinctive properties of Japanese enterprise software development, there is no past research focused on the relationship between outsourcing ratio and productivity of software development. One of our analysis results

showed that outsourcing ratio had a moderate relation to productivity. Higher outsourcing ratio projects had lower productivity. High outsourcing ratio would yield management overhead.

Also, analyzing Japanese enterprise software development projects, we confirmed almost same results as past researches describes below:

- Many past researches have revealed that team size has a strong relationship to productivity [3][5][11][14]. Also in Japanese software projects, the average team size had strong relationship to productivity. This result is considered country independent since the same result was shown in past researches with other countries' projects. It is natural that the large team size causes low productivity due to communication and management overhead.
- Both our analysis and past researches showed that the business sector had moderate relationship to productivity:
  - Manufacturing projects had higher productivity.
  - Finance/insurance projects had lower productivity.

  This may be because the requirement level of security and reliability is different between these business sectors, and it is not affected by cultural difference.

6.2. Managerial and research implications

Our result showed that average team size, architecture, business sector, outsourcing ratio, programming language, duration, and FP had a relation to productivity. Below describes possible interpretations for the results and gives our suggestions for software managers who plan or run a project.

- Average team size: Average team size had a strong relation to productivity. This result suggests that large team size would cause communication and management overhead. The average team size inevitably becomes large when a company needs to develop

software in a shorter period than usual, thus, software companies must be aware that the shortened development duration yields low productivity, which means the increase of total development effort (cost). Project managers should take into account the balance of delivery date and cost when deciding a project plan.

- Architecture: Intranet/web architecture projects showed low productivity. One interpretation is that intranet/web systems are often developed to replace old infrastructure systems. A project manager needs to consider the decrease of productivity for such a replacement project.
- Business Sector: Projects for the finance/insurance had the lowest productivity. It is expected that finance/insurance systems require a higher level of security and reliability as they deal with customer data and money. A project manager must be aware of this fact.
- Outsourcing ratio: High outsourcing ratio projects had worse productivity than low outsourcing ratio projects. Especially, for large FP projects, outsourcing ratio greatly influenced the productivity. The manager needs to consider the trade-off between loss of productivity and savings in terms of staffing costs through the use of outsourcing.
- Programming Language: COBOL projects had the lowest productivity. Interestingly, productivity of C projects became as low as COBOL in large projects. The manager needs to be careful when estimating the required effort for large C projects.

In addition to the implication on the project management above, our results also give some implications on the company management as follows:

- When software companies consider improving their own performance, they first need to recognize their performance level compared to other companies. Using our results as organizational benchmarks, companies can recognize the deviation from average performance for each project attribute. For example, referring to Figure 1, Figure 5 and Figure 12, companies can recognize their performance levels for different software

architecture styles. In addition, based on these Figures, companies could decide their goals of performance and this would promote their process improvement activities.

- Our analysis revealed possibly controllable attributes (team size, outsourcing ratio, project duration and programming language) that relate to the productivity. This result shows which attributes companies should focus on when building performance improvement plans. A straightforward way to improve productivity is to develop software with smaller team size, lower outsourcing ratio, shorter project duration or not using COBOL language. However, obviously there are some trade-offs among these attributes, e.g. smaller team size yields longer project duration. Therefore, companies must take such trade-offs into account when building plans.
- To learn from past software projects and keep improving software development processes, companies need to continuously record and accumulate various metrics of their finished projects. However, since data recording itself requires significant effort, many companies do not record enough data. On the other hand, some companies record so many metrics but not well used. To encourage data recording and use, our recommendation is to focus on project attributes that showed significant relationship to productivity. Since these attributes can be directly used to productivity evaluation and improvement, companies would be more motivated in recording and analyzing these attributes.

6.3. Limitations and future research perspectives

There are some limitations in our analysis. One limitation is that we did not analyze required and resultant software quality due to a lack of sufficient data. In the future study, we will include an analysis of quality since higher quality products may require more development effort, especially in the testing phases. Another limitation of our study is the lack of analysis of the staffing costs of developers. Further analysis is needed to clarify the trade-off between loss

of productivity and savings in terms of staffing costs through the use of outsourcing.

7. Conclusions

This paper analyzed factors of software development productivity of Japanese software enterprise applications using a software engineering data repository established by the Software Engineering Center (SEC), Information-technology Promotion Agency, Japan. Our findings include the followings:

- Average team size had a strong relationship to productivity (the variance explained was 0.39). A larger average team size was associated with lower productivity.
- Architecture also had a strong relationship to productivity (the variance explained was 0.22). Stand alone projects had higher productivity, and intranet/web projects had lower productivity.
- Business sector had a moderate relationship to productivity (the variance explained was 0.19). Manufacturing projects had higher productivity, and finance/insurance projects had lower productivity.
- Outsourcing ratio, which is a controllable attribute, also had a moderate relationship to productivity (the variance explained was 0.16). Projects with lower outsourcing ratios had higher productivity.
- Project duration and programming language showed a weak relationship with productivity. A longer duration was associated with lower productivity. Visual Basic and Java projects had higher productivity, and COBOL projects had lower productivity.
- FP also showed a weak relationship with productivity.
- The type of system user and the use of an ERP package showed no significant relationship with productivity.

These results revealed possibly controllable attributes that relate to the productivity, which

should be focused on when building performance improvement plans. Using our results as organizational benchmarks, companies can evaluate their performance level by recognizing the deviation from average performance for each project attribute. We recommend companies to continuously record and accumulate these attributes to learn from past projects and keep improving software development processes.

Acknowledgments The authors would like to thank the Software Engineering Center, Information-technology Promotion Agency, Japan, for offering the SEC dataset. This study was supported by the EASE (Empirical Approach to Software Engineering) project of the Comprehensive Development of e-Society Foundation Software program of the Ministry of Education, Culture, Sports, Science and Technology of Japan. This work is being conducted as a part of Stage Project, the Development of Next Generation IT Infrastructure, supported by Ministry of Education, Culture, Sports, Science and Technology.

References

[1] C. Andersson, A Replicated Empirical Study of a Selection Method for Software Re-liability Growth Models, Empirical Software Engineering, 12(2) (2007) 161-182.

[2] C. Andersson, P. Runeson, A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems, IEEE Trans. on Software Eng., 33(5) (2007) 273-286.

[3] J. Blackburn, G. Scudder, and L. Wassenhove, Improving Speed and Productivity of Software Development: A Global Survey of Software Developers, IEEE Trans. on Software Eng., 22(12) (1996) 875-885.

[4] W. Conover, *Practical Nonparametric Statistics (Third Edition)* (John Wiley & Sons, New York, 1998).

[5] S. Conte, H. Dunsmore, and V. Shen, *Software Engineering Metrics and Models* (The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1986).

[6] M. Cusumano, A. MacCormack, C. Kemerer, and B. Crandall, Software Development Worldwide: The State of the Practice, IEEE Software 20(6) (2003) 28-34.

[7] L. Duvall, A study of software management: The state of practice in the United States and Japan, Journal of Systems and Software 31(2) (1995) 109-124.

[8] M.Y. Huh, K. Song, DAVIS: A Java-based Data Visualization System, Computational Statistics 17(3) (2002) 411-423.

[9] A. Inselberg, The plane with parallel coordinates, The Visual Computer 1(4) (1985) 69-91.

[10] C. Jones, *Software Assessments, Benchmarks, and Best Practices* (Addison Wesley, Reading, MA, 2000).

[11] K. Maxwell, L. Wassenhove, and S. Dutta, Software Development Productivity of European Space, Military, and Industrial Applications, IEEE Trans. on Software Eng. 22(10) (1996) 706-718.

[12] K. Maxwell, and P. Forselius, Benchmarking Software Development Productivity, IEEE Software 17(1) (2000) 80-88.

[13] E. Mendes, and C. Locan, Replicating Studies on Cross- vs Single-company Effort Models Using the ISBSG Database, Empirical Software Engineering, 13(1) (2008) 3-37.

[14] P. Pendharkar, J. Rodger, An Empirical Study of the Impact of Team Size on Software Development Effort, Information Technology and Management 8(3) (2007).

[15] R. Premraj, M. Shepperd, B. Kitchenham, and P. Forselius, An Empirical Analysis of Software Productivity over Time, *Proceedings of 11th IEEE International Software Metrics Symposium (METRICS'05)*, Como, Italy, September (2005) 37.

[16] Software Engineering Center, Information-technology Promotion Agency, Japan, *The 2005 White Paper on Software Development Projects (in Japanese)* (Nikkei Business Publications, Tokyo, Japan, 2005).

[17] Software Engineering Center, http://www.ipa.go.jp/english/sec/

[18] M. Tsunoda, A. Monden, H. Yadohisa, N. Kikuchi, and K. Matsumoto, Productivity Analysis of Japanese Enterprise Software Development Projects, *Proceedings of 4th International Workshop on Mining Software Repositories (MSR 2006)*, Shanghai, China, May (2006) 14-17.

[19] B. Winer, D. Brown, and K. Michels, *Statistical Principles in Experimental Design (3rd edition)* (McGraw-Hill, New York, 1991).

Table 1 Productivity statistics

Mean	Median	Variance	Min.	Max.
0.15	0.12	0.014	0.01	0.73

Table 2 Project attributes used in the analysis

Attribute	Scale	Percentage of missing values	Description
Business sector	Nominal	31.3%	Manufacturing, Communications, Wholesale/Retail, Finance/Insurance or Service
Programming language	Nominal	39.3%	COBOL, Visual Basic, Java or C
Architecture	Nominal	0.0%	Stand Alone, Mainframe, 2-layer Client/Server, 3-layer Client/Server or Intranet/Web
System user	Nominal	0.0%	Specified or Not Specified
Use of ERP package	Nominal	14.7%	Yes or No
FP	Ratio	0.0%	Function points
Project duration	Ratio	0.0%	Duration between project start and software release dates
Average team size	Ratio	0.0%	$\text{Average team size} = \frac{\text{Effort (Person month)}}{\text{Project duration}}$
Outsourcing ratio	Ratio	62.6%	$\text{Outsourcing ratio} = \frac{\text{Outsourced effort}}{\text{Total effort}}$

Table 3 Thresholds for ratio scale attributes

Attributes	**Lower quartile ("Low" group)**	**Higher quartile ("High" group)**
Outsourcing ratio	0.60	0.86
Project duration	4.02	11.00
Average team size	2.31	9.14
FP	306	1292

Table 4 ANOVA results

Attribute	Variance explained (ω^2)	***p*-value**
Average team size	0.39	*0.000*
Architecture	0.22	*0.000*
Business sector	0.19	*0.000*
Outsourcing ratio	0.16	*0.001*
Project duration	0.08	*0.000*
Programming language	0.07	*0.006*
FP	0.05	*0.002*
System user	0.00	0.548
Use of ERP package	-0.01	0.737

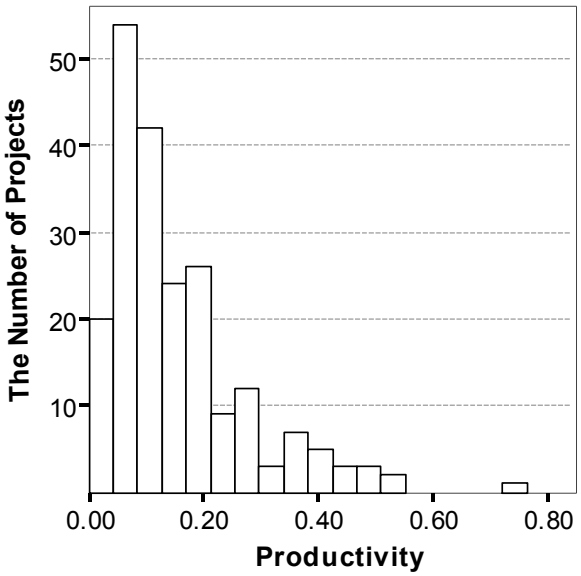

Fig. 1 Productivity distribution

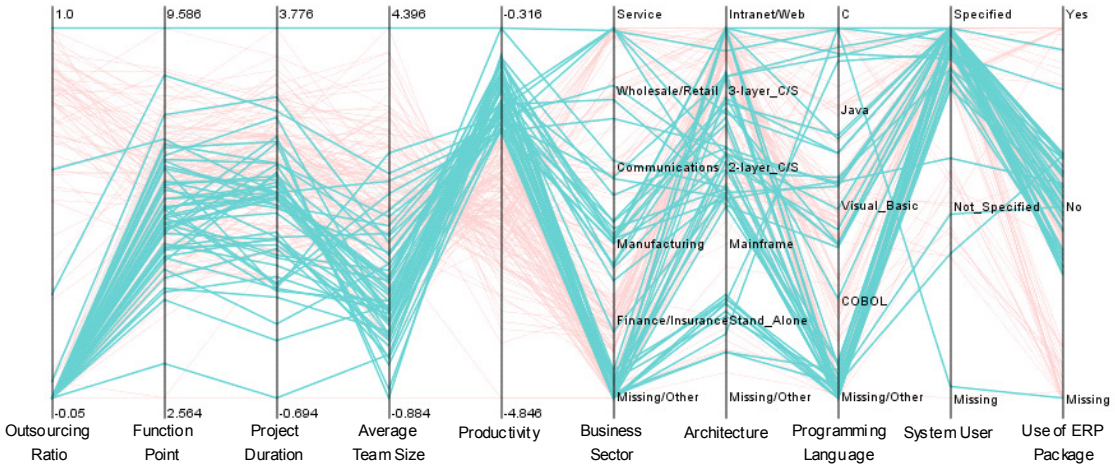

Fig. 2 PCP for high productivity projects

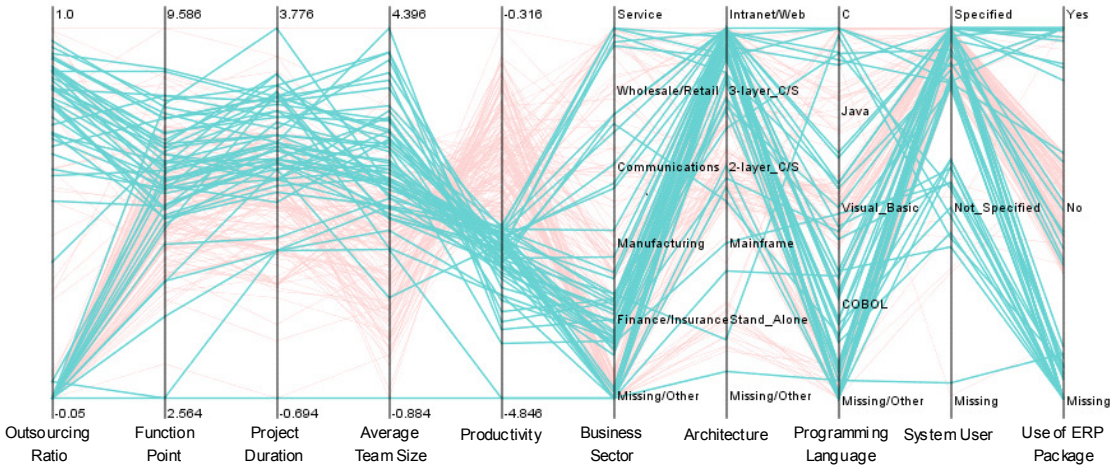

Fig. 3 PCP for low productivity projects

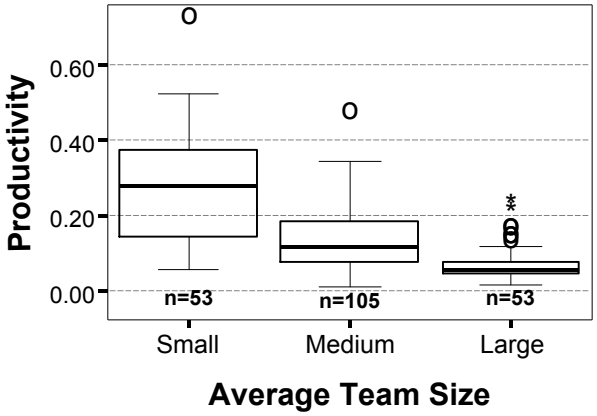

Fig. 4 Relationship between average team size and productivity

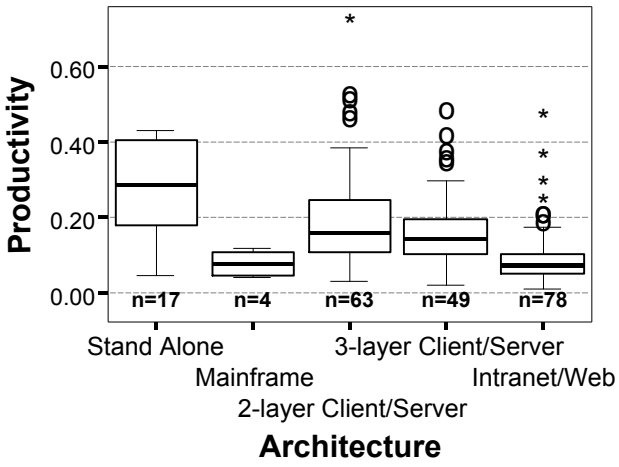

Fig. 5 Relationship between architecture and productivity

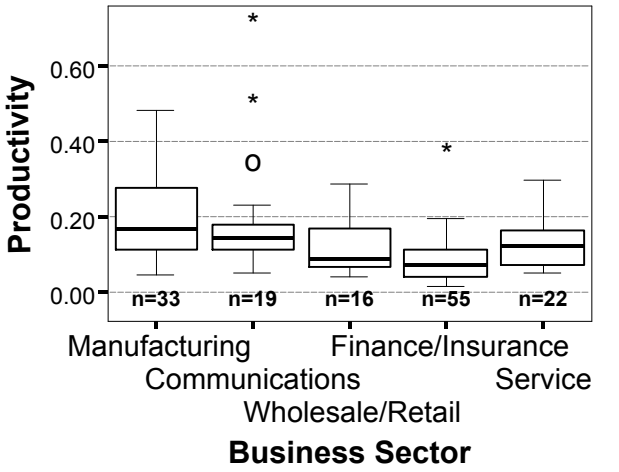

Fig. 6 Relationship between business sector and productivity

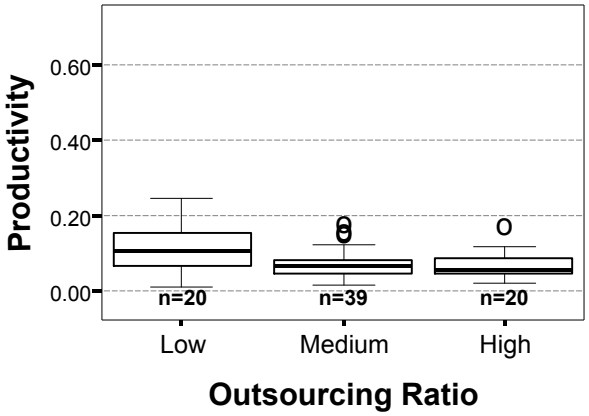

Fig. 7 Relationship between outsourcing ratio and productivity

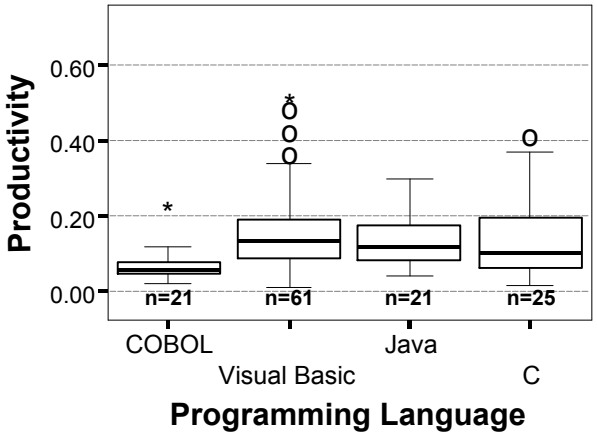

Fig. 8 Relationship between programming language and productivity

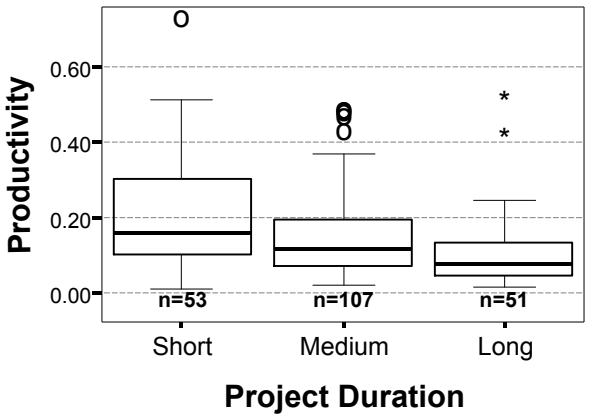

Fig. 9 Relationship between project duration and productivity

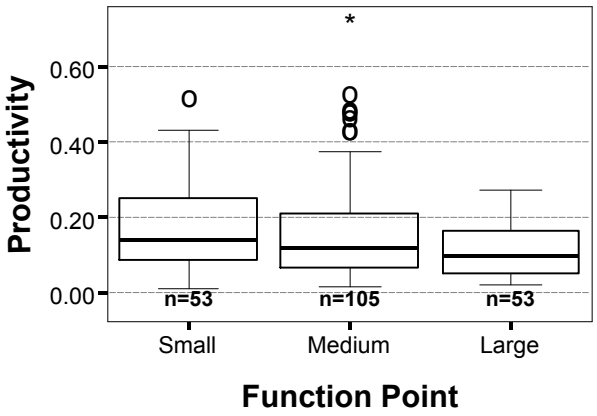

Fig. 10 Relationship between FP and productivity

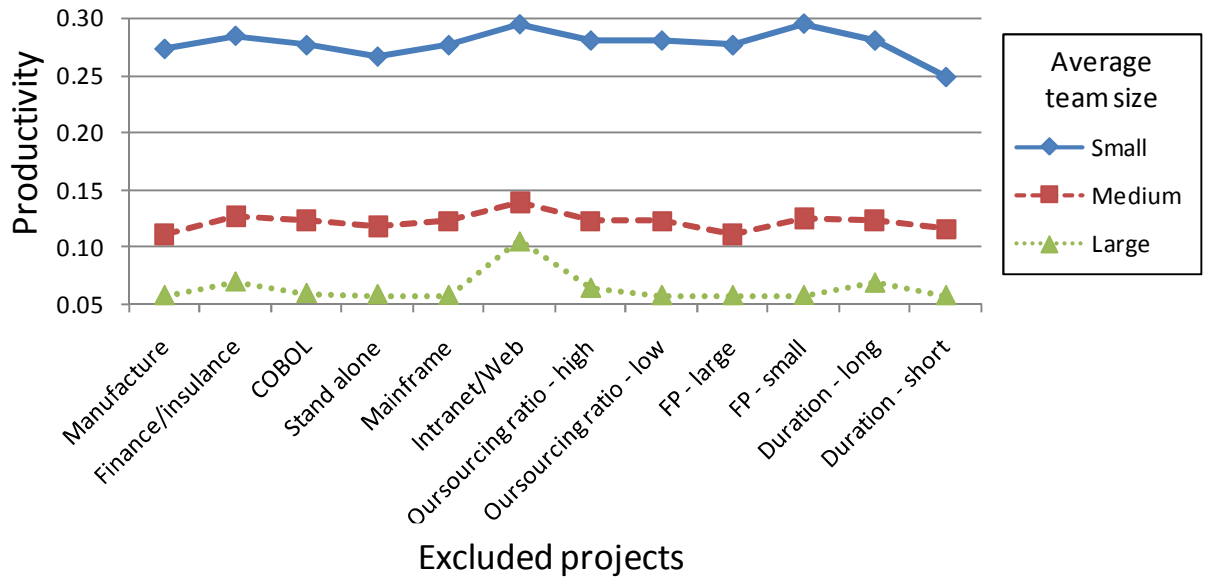

Fig. 11 Median of productivity of projects classified by average team size, excluding specific projects.

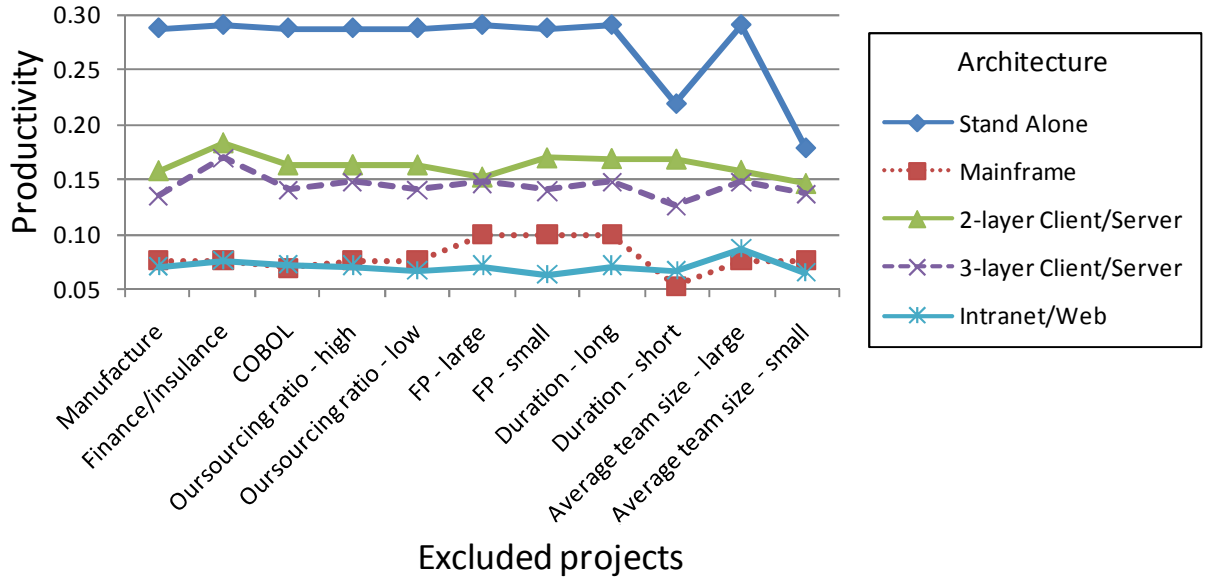

Fig. 12 Median of productivity of projects classified by architecture, excluding specific projects.

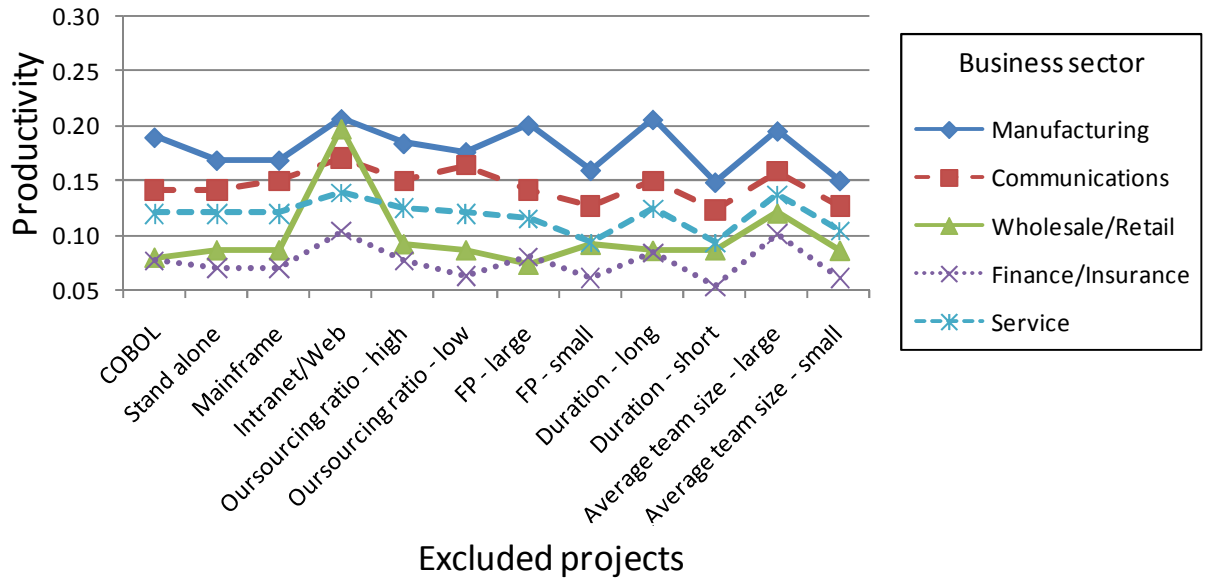

Fig. 13 Median of productivity of projects classified by business sector, excluding specific projects.

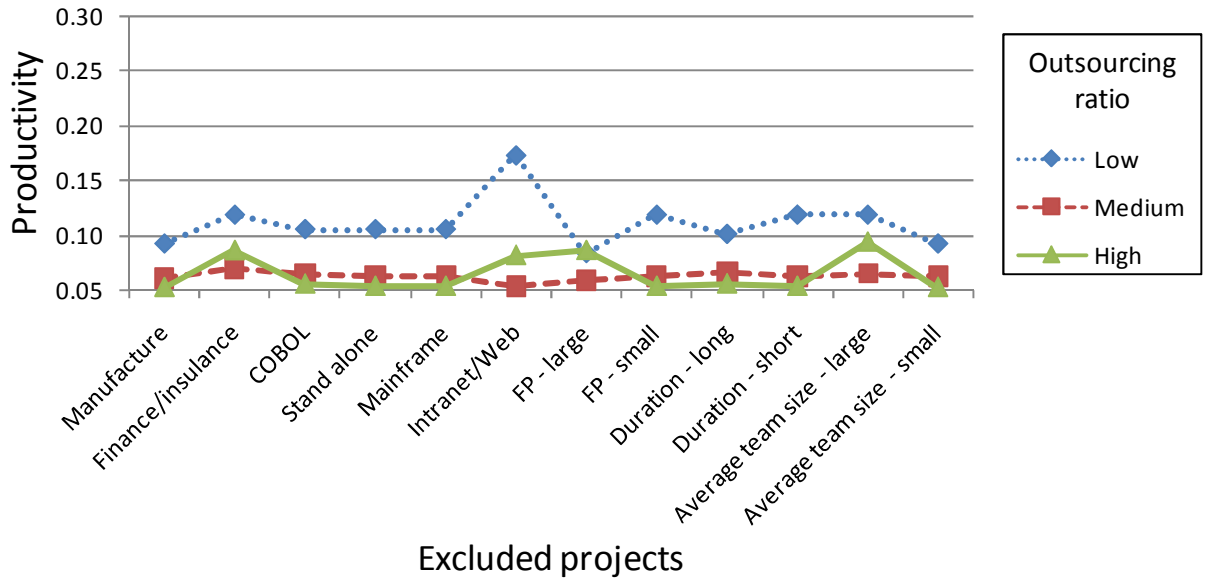

Fig. 14 Median of productivity of projects classified by outsourcing ratio, excluding specific projects.

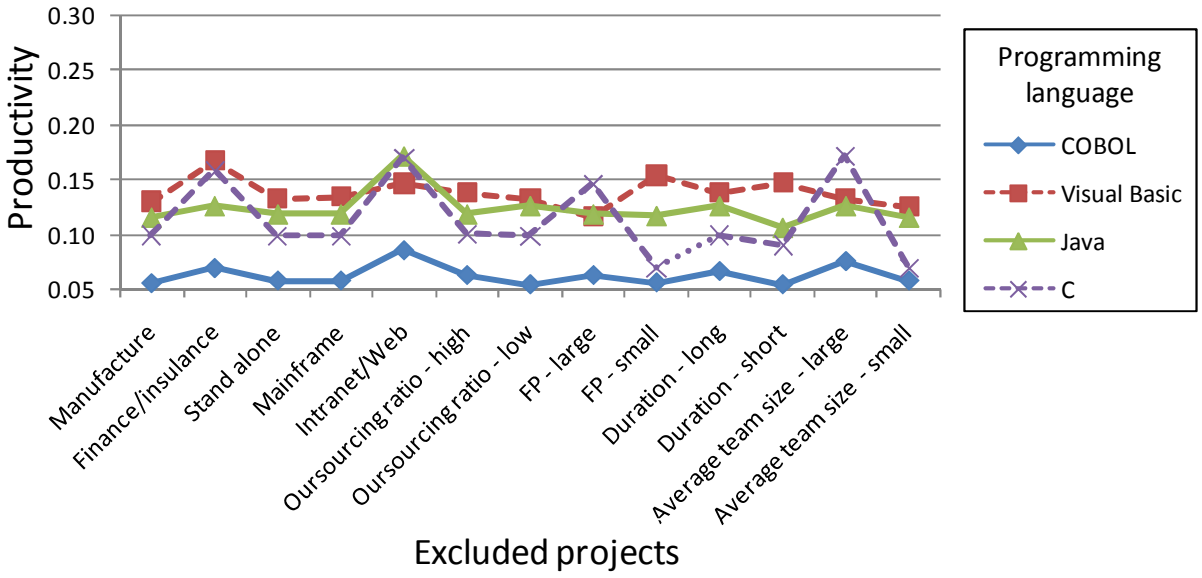

Fig. 15 Median of productivity of projects classified by programming language, excluding specific projects.

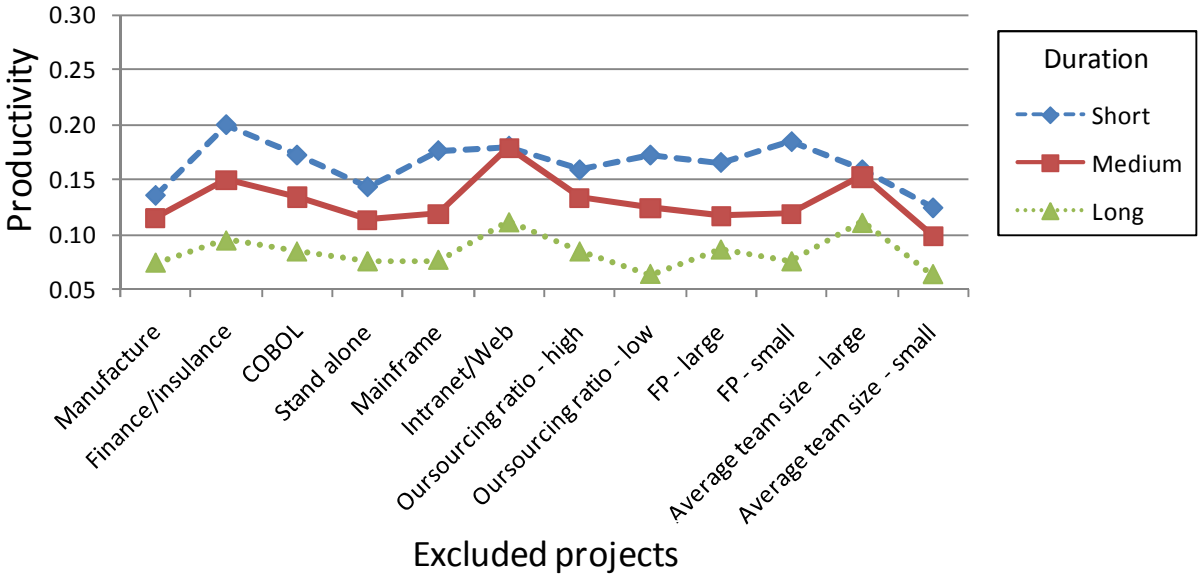

Fig. 16 Median of productivity of projects classified by project duration, excluding specific projects.

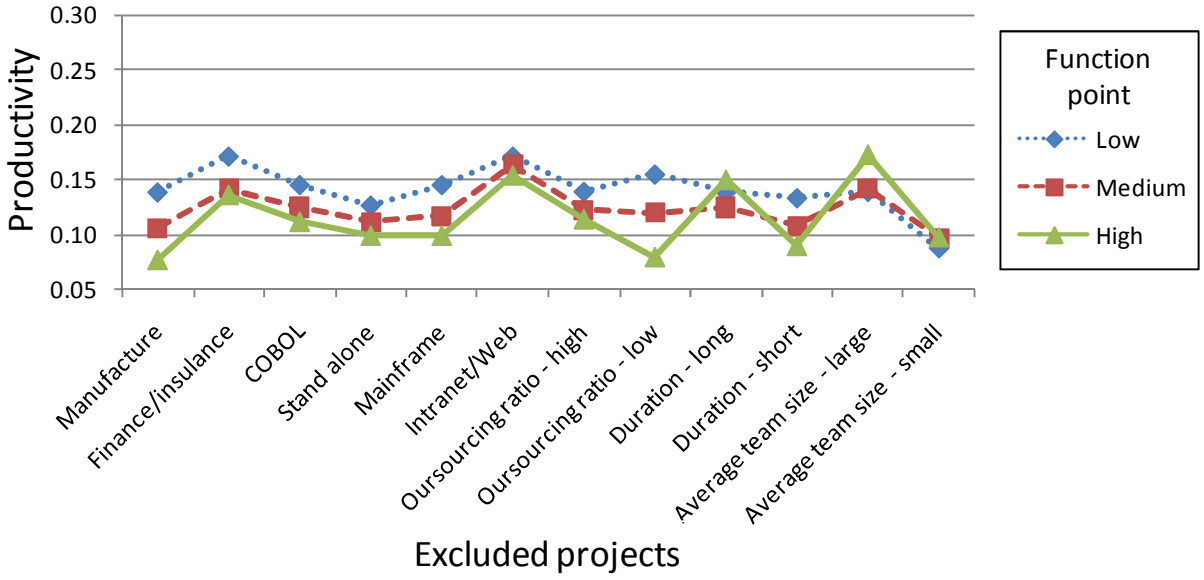

Fig. 17 Median of productivity of projects classified by FP, excluding specific projects.

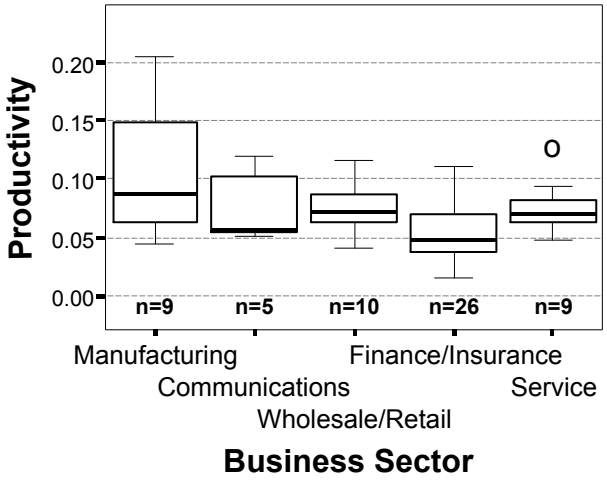

Fig. 18 Relationship between business sector and productivity, selecting only intranet/web architecture projects.

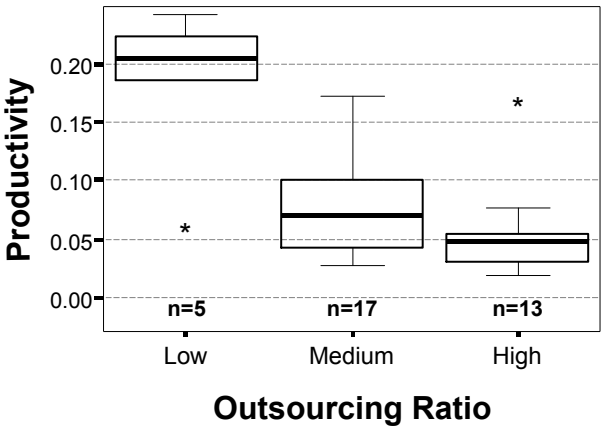

Fig. 19 Relationship between outsourcing ratio and productivity, selecting only large FP projects.

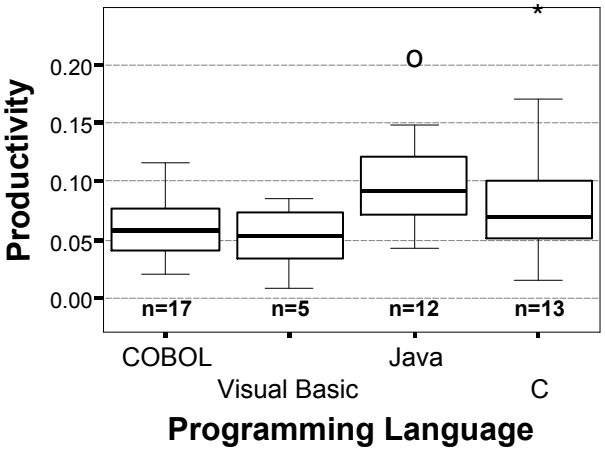

Fig. 20 Relationship between programming language and productivity, selecting only intranet/web architecture projects.