e-Informatica Software Engineering Journal, Volume 12, Issue 1, 2018, pages: 27–49, DOI 10.5277/e-Inf180102

An Empirical Study on the Factors Affecting Software Development Productivity

Luigi Lavazza[*], Sandro Morasca[*], Davide Tosi[*]
[*]*Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria*
luigi.lavazza@uninsubria.it, sandro.morasca@uninsubria.it, davide.tosi@uninsubria.it

Abstract

Background: Software development productivity is widely investigated in the Software Engineering literature. However, continuously updated evidence on productivity is constantly needed, due to the rapid evolution of software development techniques and methods, and also the regular improvement in the use of the existing ones.
Objectives: The main goal of this paper is to investigate which factors affect productivity. It was also investigated whether economies or diseconomies of scale exist and whether they may be influenced by productivity factors.
Method: An empirical investigation was carried out using a dataset available at the software project repository ISBSG. The major focus was on factors that may affect productivity from a functional point of view. The the conducted analysis was compared with the productivity data provided by Capers Jones in 1996 and 2013 and with an investigation on open-source software by Delorey et al.
Results: This empirical study led to the discovery of interesting models that show how the different factors do (or do not) affect productivity. It was also found out that some factors appear to allow for economies of scale, while others appear to cause diseconomies of scale.
Conclusions: This paper provides some more evidence about how four factors, i.e., programming languages, business areas, architectural types, and the usage of CASE tools, influence productivity and highlights some interesting divergences in comparison with the results reported by Capers Jones and Delorey et al.

Keywords: effort, function point, empirical study, ISBSG dataset, factors, development, productivity

1. Introduction

Productivity is one of the crucial aspects in software development, as it is intrinsically related to software costs. Improvements in software development productivity may come from the industrial use of novel techniques constantly introduced in Software Engineering. Also, software development productivity may improve because of the ever increasing knowledge and experience acquired on existing software engineering techniques, which, in addition, are becoming more and more consolidated over time. However, it needs to be checked if this potential improvement in productivity actually takes place and, if so, to what extent and under what conditions, so that conditions favoring productivity can be created and maintained in the software industry.

Many factors are believed to significantly influence productivity [1], so identifying relationships between factors and productivity is no simple matter. In addition, Software Engineering is still a relatively recent discipline and its empirical laws still need to be accurately described and validated. Moreover, Software Engineering is very human-intensive, thus productivity is certainly affected by factors that may not be easy to quantify and control. The human-intensive nature of software development may also imply that there are intrinsic limits to potential improvements in productivity.

This paper reports on an empirical study which was carried out to investigate whether and to what extent productivity is influenced by a number of factors, namely, the primary programming language used to develop each software project, the business area addressed by the project, the architectural type adopted by the project, and the use of CASE (Computer-Aided Software Engineering) tools. It was also investigated whether economies or diseconomies of scale (i.e. the cost disadvantages that companies accrue due to an increase in company size or output resulting in the production of services at increased per-unit costs) may exist and whether they depend on the factors that influence productivity.

The data used in this empirical study came from projects in the ISBSG (International Software Benchmarking Standards Group)[1] dataset [2], one of the most extensive datasets containing data on software development projects, and especially effort data, spanning 25 years. The ISBSG dataset contains data from a few thousand projects. Even though this is a fairly large amount of data, the ISBSG dataset represents a limited sample of the software development projects that have been and still are being carried out worldwide. Moreover, its data are provided on a voluntary basis by different types of software developers. As a result, ISBSG data may be only partially representative of all current software development practices. At any rate, ISBSG data are about projects with the same or similar characteristics as a fairly large part of current software development projects.

The main focus was on productivity from a functional point of view, so the functional size of product is measured (in Function Points [3,4]), rather than the physical size (e.g. measured in Lines of Code – LoC).

The set of factors investigated in this paper extends the set of factors studied in the authors' previous work [5], in which they were only interested in understanding the effect of the primary programming language on software productivity. In the work documented in this paper, more factors are investigated, as described in the following research question.

RQ1: Which factors influence productivity? Specifically: Does the primary programming language factor affect productivity (i.e. does productivity increase or decrease with the adopted programming language)? Does the business area factor affect productivity? Does the type of architecture factor affect productivity? Does the use of CASE tools affect productivity?

Also, the following additional research question, related to whether a factor may determine software development economies or diseconomies of scale, are addressed here.

RQ2: Which factors influence economies and diseconomies of scale? Specifically: Does the choice of the primary programming language determine a relation between size and development effort characterized by economies (or diseconomies) of scale? Similarly, do the business area, the type of architecture or the use of CASE tools determine a relation between size and development effort characterized by economies (or diseconomies) of scale?

Several different analyses were carried out. Firs a "naïve" analysis was carried out, by looking at the mean, median, and variance of the productivity for the projects in the ISBSG dataset and assessing differences across different subsets of projects, grouped according to the programming language and the other factors mentioned above. Then the productivity level of each programming language was compared with the data reported by Capers Jones [6, 7] and Delorey et al. [8], to investigate whether our productivity data are aligned with these reference data. To investigate the existence of economies and diseconomies of scale, regression models that correlate size and effort for each value of a productivity-influencing factor were built to highlight the dependence of productivity on size [1, 9] (see Section 7). All of the analyses done in the paper address both the complete ISBSG

[1]Most of the Repository Field Descriptions of the ISBSG dataset are available at: http://isbsg.org/2016/04/06/what-you-can-find-in-the-2016-r1-isbsg-development-enhancement-repository/.

data set and the “new development” and “enhancement” projects subsets separately.

The main contributions of our work with respect to the existing literature mainly lie in the fact that our study:

– is based on the analysis of a large, public dataset, namely the ISBSG dataset;
– provides up-to-date indications by analyzing recent software project data;
– addresses several factors that are believed to affect productivity;
– uses a rigorous statistical approach.

The remainder of the paper is organized as follows: Section 2 describes the analysis method used. Sections 3–6 report the analysis of productivity versus the considered factors. Section 7 discusses how each factor may contribute to the software development of economies of scale or diseconomies in software development. Section 8 lists possible threats to the validity of this work. Section 9 reviews related work. The conclusion are presented in Section 10.

2. Analysis method

2.1. Software development productivity

The adopted definition of productivity was very simple: the functional size of software developed divided by the amount of effort employed in the development process.

$$\text{Productivity} = \frac{\text{Size of developed software}}{\text{Software development effort}}$$

In this paper, the preferred size measures are the functional ones, mainly the Unadjusted Function Points (UFP) [4], although occasionally the lines of code (LoC) were used to compare these findings with those of other authors who used LoC measures. The amount of effort spent on developing software is given by the total number of person-hours or person-months spent in the development process.

2.2. The ISBSG dataset

The study reported here is based on the analysis of data from the ISBSG dataset release R12 [2]. The ISBSG dataset supports the definition of productivity given above. Specifically, many of the projects in the ISBSG dataset were measured by means of IFPUG (International Function Point Users Group) Function Points [4] or other essentially equivalent functional size measures, like NESMA (Netherlands Software Metrics users Association) Function Points [10]. The ISBSG dataset also contains development effort data, normalized to take into account possible differences in development processes.

The ISBSG dataset provides several product and process measures and characteristics that can be useful in a productivity study [11]. Among these, the programming language, the business area, the architecture and the usage of CASE tools are considered and analysed as factors that may affect productivity in this paper.

To study the effect of these factors on productivity, the authors selected and grouped data samples concerning projects with the same programming language, business area, architecture, or decision whether to use CASE tools.

2.2.1. New developments vs. enhancements

The ISBSG dataset contains data concerning both new developments and enhancements of software projects. To deal with enhancements, it is necessary to take into account the following issues.

– The size of an enhancement is defined differently than the size of development from scratch, as their measurement processes are different [12].
– The size of an enhancement in Function Points actually measures the size of the part of application in which the change occurs, not the size of the change [4, 12]. For instance, the introduction ofa new transaction has the same size as making a small change in an existing transaction, provided that the two transactions have the same complexity.
– A model stating that *Effort = f(functional size of the enhancement)* is, therefore, a simplification since enhancement effort depends on both the size of the change and the overall size of the product being changed. For

Table 1. New development projects from the ISBSG dataset: descriptive statistics

	Size [UFP]	Effort [PH]	Productivity [UFP/PH]
Mean	616	6766	0.176
Median	322	3226	0.110
Stdev	776	10497	0.253
Min	51	320	0.006
Max	7400	134211	3.960

Table 2. Enhancement projects from the ISBSG dataset: descriptive statistics

	Size [UFP]	Effort [PH]	Productivity [UFP/PH]
Mean	293.7	4073.5	0.1
Median	167.5	2188	0.079
Stdev	403.7	6479.6	0.1
Min	50	322	0.004
Max	7134	109271	1.5

instance, after an enhancement, a system test must be carried out, and the effort required for this type of testing is related to the entire application size, rather than the size of the enhancement alone. Unfortunately, building a model of the *Effort=f(functional size of the application, functional size of the enhancement)* type is not possible, since the ISBSG database does not provide the sizes of the enhanced applications,only the size of the enhancements.

Because of the differences in the development from scratch and enhancement processes, the effects of programming languages, business areas, architectural types, and usage of CASE tools are investigated on new developments and enhancement projects separately.

2.2.2. Data selection

Not all ISBSG projects were suitable for this analysis. Data samples were selected according to the following criteria:

– Only projects measured in IFPUG or NESMA FP and provided with both size and effort data were selected.
– Only data concerning projects with a specified primary programming language, business area, architecture, and usage of CASE tools were selected.
– The projects in the ISBSG dataset are characterized by different quality levels. The selected projects had their data quality rated ‘A’ or ‘B’, i.e., those with good quality of data in the ISBSG dataset. Similarly, the UFP rating (i.e. quality of functional size measurement) of the selected projects was ‘C’ or greater. This is consistent with the previous studies of the ISBSG dataset.
– New development projects concerning applications smaller than 50 UFP were not considered. For such small projects, it is likely that specific effects – such as the usage of simplified life cycles – can dramatically affect productivity, thus making them hardly comparable with larger projects.
– Similarly, projects greater than 10,000 UFP or requiring more than 150,000 person-hours were not considered. There were only 4 projects with such characteristics, so they can very well be considered outliers.

2.3. Descriptive statistics

2.3.1. New development projects

Out of about 6000 ISBSG projects, 989 data points concerning new developments satisfy the selection requirements described in Section 2.2.2. The descriptive statistics are given in Table 1 (where PH indicates person-hours).

Figure 1 shows the distribution of the productivity data of the selected projects (the grey diamond is the mean value). Projects with productivity greater than 1 UFP/PH are not shown, to preserve the readability of the figure.

2.3.2. Enhancement projects

Out of about 6,000 ISBSG projects, 1570 data points concerning enhancements satisfy the selection requirements described in the previous section. The descriptive statistics are presented in Table 2.

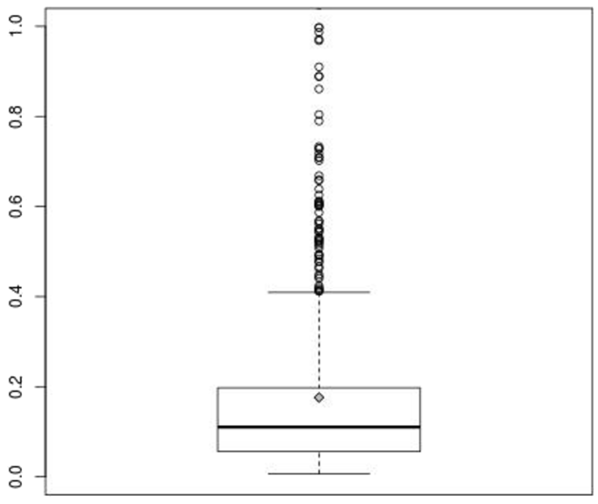

Figure 1. Distribution of productivity of new development projects

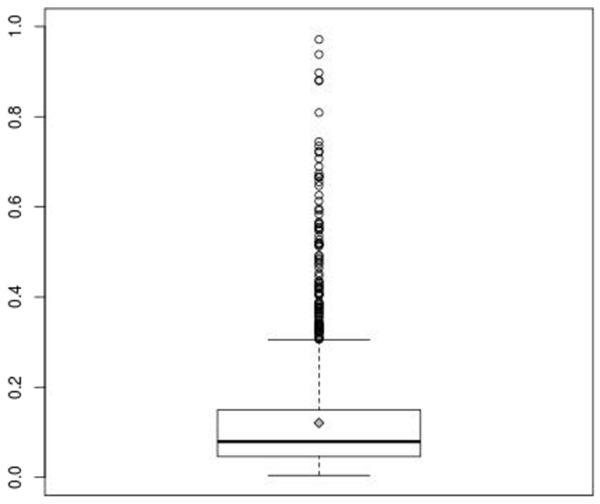

Figure 2. Distribution of productivity of enhancement projects

Figure 2 shows the distribution of the productivity data of the selected projects (the grey diamond is the mean value). Projects with productivity greater than one UFP/PH are not shown, to preserve the readability of the figure.

As the first result of the analysis, one can note that productivity varies widely (the maximum observed value is 2.250% the mean and 66,000% the minimum observed value) and that the productivity of enhancement projects tends to be lower than that of new development projects, but with a smaller variance. This may appear to be somewhat surprising, since the value of UFP for an enhancement project is the size of the part of the application where the enhancement takes place, regardless of the size of the change itself. Therefore, the result shows that, on average, more effort is used in an enhancement project than in a new development project with the same functional size. This is probably due to the fact that maintenance is more challenging than development from scratch.

2.4. Data analysis techniques

We applied several statistical data analysis techniques. The Shapiro–Wilks test was used to check whether specific distributions are normal, and the nonparametric Kruskal–Wallis [13] and Mann–Whitney tests [9] to check if a nominal independent variable affects productivity.

Power law models, i.e. models of the kind $\mathit{Effort} = e\mathit{UFP}^b$, were used to investigate whether a statistical relationship exists between UFP and Effort. Ordinary Least Square (OLS) regression techniques were used after applying logarithmic transformations to both UFP and Effort, because the assumptions about the normality of distributions do not hold for UFP and Effort. Power-law models are used to investigate the existence of economies or diseconomies of scale.

In the paper, the statistical significance threshold is set to 0.05, as customary in Empirical Software Engineering studies. All of the statistical results reported in the paper are statistically significant, i.e. they have p-value < 0.05.

3. Effects of primary programming language on productivity

The impact of the programming language primarily used to develop the project was analysed. Different programming languages call for different development processes, skills, data structures, methods, testing activities, and so on. It is thus reasonable to expect that the productivity of software development may depend on the programming language.

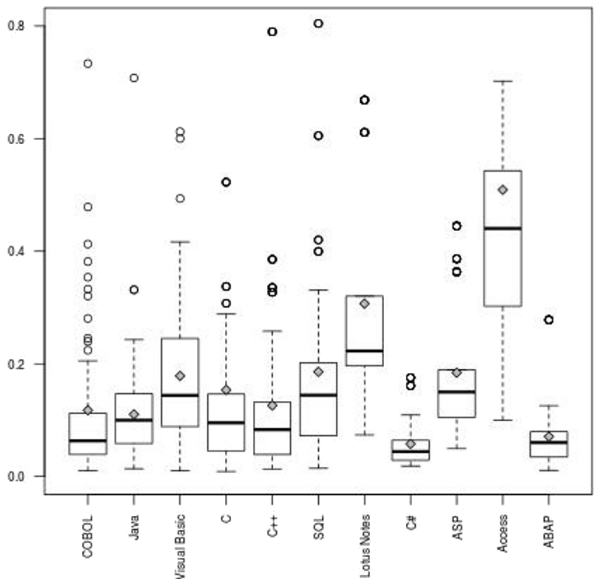

Figure 3. Distributions of new development productivity per programming language

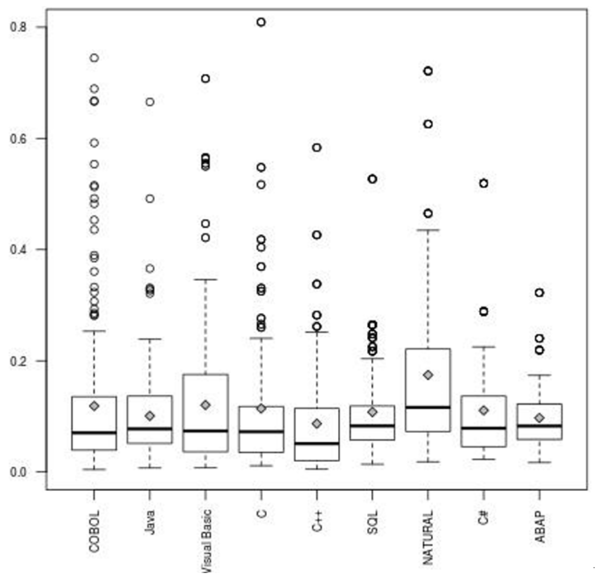

Figure 4. Distributions of enhancement project productivity per programming language

3.1. New development projects

Table 3 gives a few descriptive statistics of new development projects grouped by the programming language. The median productivity greatly changes from a minimum of 0.044 UFP/PH for C# projects to a maximum of 0.425 UFP/PH for access projects. This reinforces the idea that productivity may depend on the programming language.

The distributions of the productivity of projects grouped by language are shown in Figure 3. As the figure shows, the distributions are far from symmetrical, so the "distribution-free" nonparametric Kruskal–Wallis rank sum test [13] was used to assess whether the difference between groups is significant. The results ($\chi^2 = 291.66$, $df = 70$, p-value $< 10^{-15}$) confirm that the primary programming language has a significant effect on productivity.

The authors proceeded to study the effect of the programming languages on productivity for pairs of different programming languages, using the Mann–Whitney test, to check if there was a statistically significant order relationship between the subsets of projects with different pairs of languages. The results are reported in Table 4. The symbol '>' denotes that the projects with the programming language reported in the row of a cell have higher productivity (in a statistically significant sense) than those with the programming reported in the column. Likewise, the symbol '<' denotes the opposite relationship. The symbol '=' denotes that no statistically significant difference was found.

The projects based on the language used in Access appear to be the most productive ones, followed by those based on Lotus Notes. Surprisingly, the productivity of C# projects appears to be the worst one, followed by ABAP, and C++. There is no empirical evidence on the reasons why Access appears very productive while the productivity of C# development appears very low. It can be argued that high-level languages, such as Access, are more productive since they are used in simpler projects and business processes than more complex languages (such as C#) that are generally used in more complex projects of several kinds of application areas.

3.2. Enhancement projects

Table 5 gives a few descriptive statistics for enhancement projects, grouped by their programming language. Note that the languages that appear in Table 5 are not the same as those appearing in Table 3, because in Table 5 the

Table 3. Summary data of new development projects grouped by programming language

Language	N	Median Size [UFP]	Median Effort [PH]	Median [UFP/PH]
COBOL	174	286	4333.5	0.063
Java	114	281.5	3394.5	0.103
Visual Basic	145	327	2760	0.145
C	48	479	4712	0.098
C++	37	312	5100	0.083
SQL	48	615.5	5662.5	0.144
Lotus Notes	16	275.5	1117	0.223
C#	22	285.5	6859.5	0.044
ASP	14	282.5	1957	0.150
Access	24	359.5	845	0.425
ABAP	17	279	6051	0.060

Table 4. Relations between new development productivity of programming languages

Language	COBOL	Java	Visual Basic	C	C++	SQL	Lotus notes	C#	ASP	Access	ABAP
COBOL		<	<	<	=	<	<	>	<	<	=
Java	>		<	=	=	<	<	>	<	<	>
Visual Basic	>	>		>	>	=	<	>	=	<	>
C	>	=	<		=	=	<	>	<	<	>
C++	=	=	<	=		<	<	>	<	<	=
SQL	>	>	=	=	>		<	>	=	<	>
Lotus Notes	>	>	>	>	>	>		>	=	<	>
C#	<	<	<	<	<	<	<		<	<	=
ASP	>	>	=	>	>	=	=	>		<	>
Access	>	>	>	>	>	>	>	>	>		>
ABAP	=	<	<	<	=	<	<	=	<	<	

languages with too few data to support any statistically significant analysis were omitted. The median productivity varies much less than for new development projects, from a minimum of 0.051 UFP/PH for C++ projects to a maximum of 0.116 UFP/PH for NATURAL projects, with the next ones equal to 0.083 UFP/PH for SQL, C#, and ABAP projects. Thus, productivity may depend less on the programming language for enhancement than for new developments.

The distributions of the productivity of projects grouped by programming language are shown in Figure 4.

The comparison of Figures 3 and 4 seems to confirm that the productivity of enhancement projects appears much less dependent on programming languages than the productivity of new development projects. Moreover, it appears that for several languages the productivity in enhancements is substantially lower than the productivity of new developments. The projects based on the NATURAL language [14] are associated with higher productivity than the projects based on other languages. However, in Table 6 the sign '=' occurs more frequently than in Table 4, indicating that the productivities of several language are statistically not discriminated in enhancement projects.

3.3. Comparison with Capers Jones productivity evaluations

Capers Jones [6] studied the relation between the language "level" and its productivity [6]. The language level is defined according to the LoC/FP ratio: the larger the number of lines of code needed to code a Function Point, the lower the level of the language. For example, COBOL requires about 105 statements per FP and is classified as a level 3 language [6]. Table 7 lists the average LoC per FP, the language level, and the average productivity in FP/PM (where PM denotes person-months) according to Jones. In this paper, $PM = PH/160$, where 160 is obtained by multiplying 20 working days per month and 8 working hours per day); the reported data are the result of an analysis concerning software developed up till 1996. To be able to compare our results with those by Capers Jones, the productivity of projects carried out up till 1996 was analysed separately (in columns "Pre" in the tables of this paper) and after 1996 (in columns "Post").

Descriptive statistics are given in Table 8 and Table 9.

These results seem to indicate that there has been a decrease in the productivity for both new developments and enhancements. In the opinion of the authors, the most likely cause is that software complexity has considerably grown, so

Table 5. Summary data of enhancement projects grouped by programming language

Language	N	Median Size [UFP]	Median Effort [PH]	Median Prod. [UFP/PH]
COBOL	306	179	2583	0.070
Java	271	142	2026	0.077
Visual Basic	132	217.5	3154	0.075
C	113	181	2705	0.072
C++	79	141	3810	0.051
SQL	59	142	1837	0.083
NATURAL	55	214	1694	0.116
C#	30	258.5	2728.5	0.083
ABAP	45	249	3069	0.083

Table 6. Relations between enhancement project productivity of programming languages

Language	COBOL	Java	Visual Basic	C	C++	SQL	NATURAL	C#	ABAP
COBOL		=	=	=	>	=	<	=	=
Java	=		=	=	>	=	<	=	=
Visual Basic	=	=		=	>	=	<	=	=
C	=	=	=		=	=	<	=	=
C++	<	<	<	=		<	<	<	<
SQL	=	=	=	=	>		<	=	=
NATURAL	>	>	>	>	>	>		>	>
C#	=	=	=	=	>	=	<		>
ABAP	=	=	=	=	>	=	<	<	

Table 7. Programming language productivity according to Jones (before 1996)

Language	LoC/FP	Level	Avg Productivity [FP/PM]
ABAP	16	20.0	15 to 30
Access	38	8.5	16 to 23
C	128	2.5	5 to 10
C++	53	6.0	10 to 20
COBOL	107	3.0	5 to 10
DELPHI	29	11.0	16 to 23
Java	53	6.0	10 to 20
SQL	13	25.0	30 to 50
Visual Basic	40	8.0	10 to 20

Table 8. New development projects from the ISBSG dataset: descriptive statistics (Pre: up to 1996, Post: after 1996)

	Size [UFP]		Effort [PH]		Productivity [UFP/PH]	
	Pre	Post	Pre	Post	Pre	Post
Mean	734	583	7319	6607	0.209	0.166
Median	415	303	3703	3074	0.121	0.108
Stdev	822	759	10162	10592	0.342	0.221
Min	53	51	326	320	0.01	0.006
Max	4943	7400	66600	134211	3.96	2.581

Table 9. Enhancement projects from the ISBSG dataset: descriptive statistics (Pre: up till 1996, Post: after 1996)

	Size [UFP]		Effort [PH]		Productivity [UFP/PH]	
Mean	348	290	3750	4098	0.166	0.117
Median	248	161	2104	2193	0.114	0.078
Stdev	376	406	6980	6441	0.155	0.128
Min	52	50	339	322	0.021	0.004
Max	2983	7134	61891	109271	0.939	1.51

Table 10. Comparison with Jones (before 1996)

Language	C. Jones [6]	Our analysis	
	Mean Prod. [FP/PM]	Mean Prod. [FP/PM]	Stdev/ Mean
C	5 to 10	27	226%
COBOL	5 to 10	23	130%
SQL	30 to 50	33	106%

Table 11. Comparison with Jones
(project data up to 2013)

Language	C. Jones [7] Mean Prod. [FP/PM]	Our analysis Mean Prod. [FP/PM]	Stdev/ Mean
C	5.62	16.9	99%
COBOL	6.38	15.2	100%
ABAP	7.69	12.4	50%
C++	9.68	13.8	97%
Java	9.68	14.7	66%
C#	9.88	11.7	78%
Visual Basic	13.04	21.8	76%
ASP	13.40	24.1	53%
SQL	15.92	17.3	62%

many technological and methodological advances were "absorbed" by additional difficulty. In fact, the notion of productivity is based on functional size: it is quite possible that modern software has to satisfy more non-functional requirements than old-time software (for instance of security requirements). These additional non-functional requirements certainly require some development effort, which is not explained by the sheer implementation of the required functionality.

In the ISBSG dataset, only three languages were found with enough data points to support a reasonably reliable comparison of productivity before 1996 and after 1996. The comparison – illustrated in Table 10 – is thus limited to these three languages. The columns on the right lists the so-called coefficient of variation, which is the ratio of the standard deviation to the mean, respectively.

Table 10 shows that data from the ISBSG dataset confirm Jones's findings concerning SQL, but indicate that the mean development productivity achieved when using C or COBOL is definitely higher than that found by Jones. It can also be observed that C programming involves a great variability of the productivity level that can be achieved. This is actually not surprising, given that C was used for a wide range of applications and in very different domains.

Table 11 reports an updated set of Jones's productivity data concerning project carried out until 2013 [7].

Table 11 shows that the found mean productivities are greater than those found by Jones. Unfortunately, the authors have no means of explaining this difference. However, there are some similarities between our results and those obtained by Jones: Visual Basic, ASP and SQL appear more productive then the other languages. The main difference is that C appears quite productive according to ISBSG data, while it was ranked as the least productive language by Jones.

It is also possible to observe that the productivity of C programming was less variable after 1996 than earlier. This is probably due to the fact that after 1996 programmers could choose from among so many languages that a relatively low-level language, such as C, is used only in well characterized domains (system-level programming, real-time, etc.).

3.4. Comparison with open-source software development productivity

Delorey et al. analysed 9,999 open-source projects hosted on SourceForge.net to study the productivity of 10 of the most popular programming languages in use in the open-source community [8]. Table 12 reports the data about the languages analysed both in [8] and in this study. The central column in Table 12 provides the data derived from [8] (expressed in Function Points per PersonHour).

With respect to the study by Delorey et al., the data from the ISBSG dataset indicate much higher productivity for all languages. Although this indication is fairly consistent for

Table 12. Comparison with [8]

Language	Delorey et al. [8] Mean Prod. [FP/PH]	Our analysis Mean Prod. [FP/PH]	Our analysis Stdev/Mean
C	0.013	0.120	99%
C#	0.035	0.083	78%
C++	0.032	0.098	97%
Java	0.030	0.105	66%

Table 13. Summary data by business area for new development projects

Business area	N	Median Size [UFP]	Median Effort [PH]	Median Prod. [UFP/PH]
Engineering	18	549.5	1464.5	0.257
Accounting	19	418	4111	0.135
Financial (excl. Banking)	29	327	3123	0.125
Telecommunications	52	262.5	2574.5	0.118
Inventory	12	574	7434.5	0.114
Manufacturing	25	315	3565	0.097
Insurance	38	261	2806.5	0.087
Banking	69	214	2761	0.064

all languages, there is a noticeable difference concerning the C language: while it appears as the least productive language in [8], C appears to be the most productive according to the ISBSG data (in the set of languages considered in Table 12).

4. Effects of business areas on productivity

The previous work [1] reports that the business area can influence development productivity. Accordingly, the dependence of productivity on business areas were analysed here. Projects were thus grouped per business area and only groups of twenty or more projects were kept for statistical analysis.

4.1. New development projects

Table 13 gives the descriptive statistics of new development projects grouped by business areas. The median productivity greatly changes from a minimum of 0.064 UFP/PH for banking projects to a maximum of 0.257 UFP/PH for engineering projects – i.e. the projects supporting various types of activities (design, simulation, etc.) in various engineering areas (civil engineering, electrical engineering, etc.) – approximately four times the minimum. Thus, it can be hypothesized that productivity may depend on the business area. Quite interestingly, the low productivity of insurance projects was already detected in [1] and in [12].

The distributions of the productivity of projects grouped by business area are shown in Figure 5 (where projects with productivity greater than 1 FP/PH are not shown, to preserve the readability of the figure). Since distributions are not symmetrical, the "distribution-free" nonparametric Kruskal–Wallis rank sum test [13] was used to assess whether the difference between groups is significant. The results ($\chi^2 = 116.93$, $df = 76$, p-value $= 0.0018$) confirm that the business area has a significant effect on productivity.

Since the Kruskal–Wallis test only indicates that in at least one case the business area affects the productivity, in this research the Mann–Whitney test was used to study the effect of the business area on productivity for all pairs of different business areas.

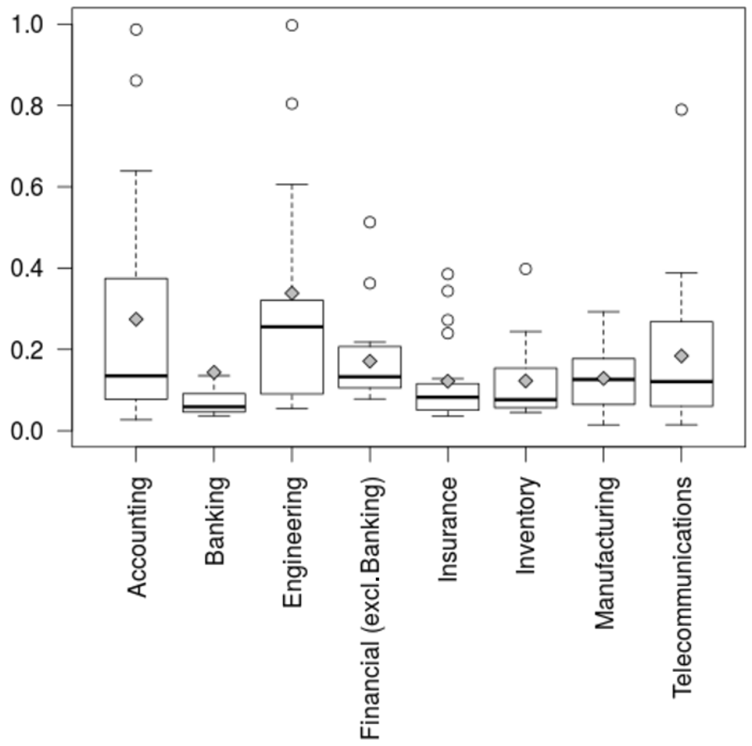

Figure 5. Distributions of new development project productivity per business area

The results of the Mann–Whitney tests are reported in Table 14, with the same conventions as the ones used in Table 6.

The projects belonging to the engineering business area appear to be the most productive ones (as for new developments), followed by those belonging to accounting and financial business areas.

4.2. Enhancement projects

Table 15 gives the descriptive statistics of enhancement projects, grouped by business area. Note that the programming languages that appear in Table 15 are not the same as those appearing in Table 13, because different numbers of data points were available for new developments and enhancement projects and, hence, the areas with too few data to support any statistically significant analysis were excluded from the analysis. The business area with the highest median productivity (Legal) has a productivity that is a bit less than five times the lowest median productivity, obtained for Quality. This suggests that productivity may depend on the business area.

The nonparametric Kruskal–Wallis method [13] was used to assess whether the difference between groups was significant. For enhancement projects, the result ($\chi^2 = 119.974$, $df = 44$, p-value $< 10^{-8}$) confirms that the business area has a statistically significant effect on productivity.

Since the Kruskal–Wallis test only indicates that in at least one case the business area affects the productivity, in these investigations the Mann–Whitney test was used to study the effect of the business area on productivity for all pairs of different business areas. The results of the Mann–Whitney tests are reported in Table 16 for enhancement projects.

On the one hand, the legal and insurance projects have the highest enhancement productivity. The insurance projects have high enhancement productivity, while they have quite low development productivity. No data were available to support this kind of analysis, but it can be argued that new insurance projects are less productive since a lot of rules and laws regulate the insurance domain. This requires a lot of effort during the initial phases of the development process, while this effort decreases over time

Table 14. Relations between productivities per business area (new developments)

Business area	Accounting	Banking	Engineering	Financial (excl. Banking)	Insurance	Inventory	Manufacturing	Telecommunications
Accounting		>	=	=	>	=	=	=
Banking	<		<	<	=	=	=	<
Engineering	=	>		=	>	>	>	>
Financial (excl. Banking)	=	>	=		>	=	=	=
Insurance	<	=	<	<		=	=	=
Inventory	=	=	<	=	=		=	=
Manufacturing	=	=	<	=	=	=		=
Telecommunications	=	>	<	=	=	=	=	

Table 15. Summary data by business area for enhancement projects

Business area	N	Median Size [UFP]	Median Effort [PH]	Median Prod. [UFP/PH]
Legal	12	419.5	1485	0.248
Insurance	38	315.5	1679	0.181
Financial (excl. Banking)	44	237.5	1881	0.112
Inbound Logistics	47	106	907	0.093
Outbound Logistics	46	120	1639	0.077
After Sales & Services	26	107	1362.5	0.076
Banking	33	198	2070	0.072
Manufacturing	47	192	3048	0.058
Quality	21	233	3487	0.051
Sales	34	190.5	2609	0.070
Telecommunications	181	142	2151	0.077

whenever legal aspects are well managed. On the other hand, the banking projects confirm their low productivity (for both new developments and enhancements).

5. Effects of architecture on productivity

Different types of architecture call for different development processes, skills and methods. It is thus reasonable to expect that development productivity depends on system architecture. Accordingly, the projects were grouped per architecture and the distributions of productivity were analysed.

5.1. New development projects

The descriptive statistics of the new development project groups characterized by the same architecture are reported in Table 19. Systems with Multi-tier/client-server architecture are characterized by the highest productivity, a bit more than twice the productivity of systems with client-server architecture, the ones with the lowest productivity.

The distributions of the productivity of projects grouped by architecture are shown in Figure 7. The differences in the boxplots do not appear to be large. Since distributions are not symmetrical, the “distribution-free” non-

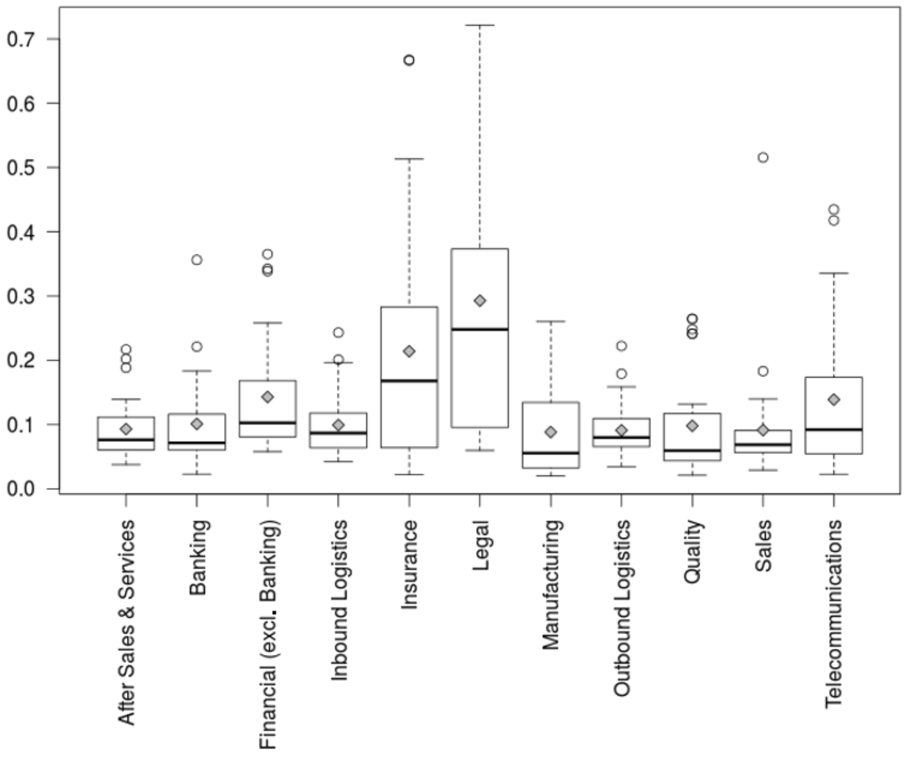

Figure 6. Distributions of enhancement project productivity per business area

Table 16. Relations between business areas (enhancements)

Business area	After Sales & Services	Banking	Financial (excl. Banking)	Inbound Logistics	Insurance	Legal	Manufacturing	Outbound Logistics	Quality	Sales	Telecommunications
After Sales Services		=	<	=	<	<	>	=	=	=	=
Banking	=		<	=	<	<	=	=	=	=	=
Financial (excl. Banking)	>	>		=	<	<	>	>	>	>	>
Inbound Logistics	=	=	=		<	<	>	=	>	>	=
Insurance	>	>	>	>		=	>	>	>	>	>
Legal	>	>	>	>	=		>	>	>	>	>
Manufacturing	<	=	<	<	<	<		<	=	=	=
Outbound Logistics	=	=	<	=	<	<	>		=	=	=
Quality	=	=	<	<	<	<	=	=		=	=
Sales	=	=	<	<	<	<	=	=	=		=
Telecommunications	=	=	<	=	<	<	=	=	=	=	

parametric Kruskal–Wallis rank sum test [13] was used to assess whether the difference between groups is significant. The results ($\chi^2 = 60.45$, *df* $= 6$, *p*-value $< 10^{-10}$) confirm that the architecture has a significant effect on productivity.

Here again, the ISBSG dataset does not provide any support for explaining, even tentatively, these results.

Since the Kruskal–Wallis test only indicates that in at least one case the business area affects the productivity, the Mann–Whitney test was used to study the effect of the architecture on productivity for all pairs of different architectures.

The results of the Mann–Whitney tests are reported in Table 18, with the same conventions as the ones used in Table 6.

5.2. Enhancement projects

Table 19 gives a few descriptive statistics of enhancement projects, grouped by architecture. The ratio between the highest and the lowest median productivity is slightly smaller than three. The distributions of the productivity of projects grouped by architecture are shown in Figure 8. The differences in the boxplots do not appear to be large.

The nonparametric Kruskal–Wallis method [13] was used to assess whether the difference between groups was significant. For enhancement projects, the results ($\chi^2 = 45.06$, *df* $= 6$, *p*-value $< 10^{-7}$) confirm that the architecture has a significant effect on productivity also for this type of projects. The effect of the architecture on productivity for pairs of different architectures was also studied using the Mann–Whitney test. The results are reported in Table 20.

Multi-tier projects are the least productive for both new development and enhancement projects, while multi-tier with web public interface projects appear to be the most productive just for enhancement projects. Multi-tier/Client server projects are the most productive for new developments, and they maintain high productivity also in the case of enhancement projects.

6. Effects of case tool usage on productivity

The use of CASE tools has long been advocated to improve the productivity of software development processes. While traditionally CASE tools were essentially diagramming/modelling tools, which adopted some sort of a semi-formal design language, such as E/R or Data Flow Diagrams, today the concept embraces all sorts of computer-based tools that are meant to support software development activities. Quite noticeably, some tools are meant to support agile development. For instance, there are tools for writing and managing user stories and tools for writing wire frames and GUI mockups, etc. So, in the ISBSG dataset, “CASE” equates to any computer-based tool supporting software development. However, it can be expected that, in most cases represented in the ISBSG dataset, the used CASE tools are traditional.

Although the usage of CASE (Computer-Aided Software Engineering) tools in software development is conceptually a Boolean variable, in the ISBSG dataset there are four possible values: Yes, No, Don’t know and Null (i.e. no value was provided). In the analysis of the effects ofCASE tool usage on productivity, the projects for which there is no clear indication of whether CASE tools were used or not were neglected. That is, only the projects having “CASE tool usage” field equal to Yes (497 projects) or No (851 projects) were retained.

As in the previous cases, the nonparametric Kruskal–Wallis [13] was used to assess whether the difference between groups was significant. The results do not support the hypothesis that the usage of CASE tools has a significant effect on productivity for either new development or enhancement projects.

This result is confirmed by the Mann-Whitney tests on pairs.

Table 17. Summary data of new development projects grouped by architecture

Business area	N	Median Size [UFP]	Median Effort [PH]	Median Prod. [UFP/PH]
Multi-tier/Client server	113	410	2519	0.184
Multi-tier with web public interface	46	169	1470	0.140
Stand alone	234	308.5	3047.5	0.114
Multi-tier	24	479	6496	0.094
Client server	223	350	4628	0.079

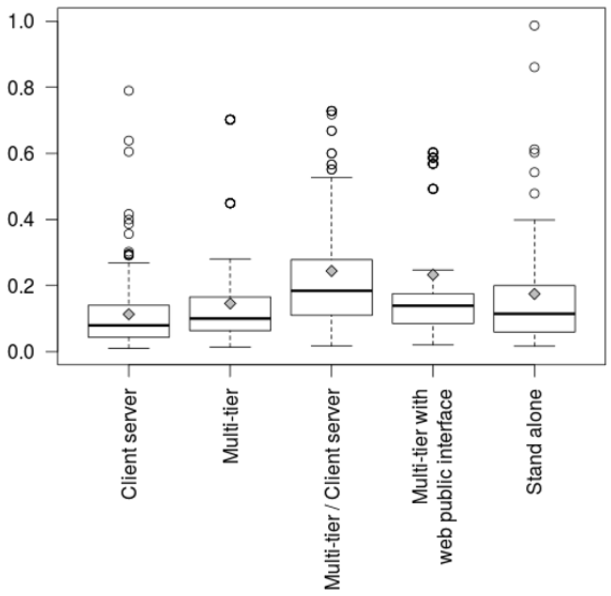

Figure 7. Distributions of productivity per architecture type (new developments)

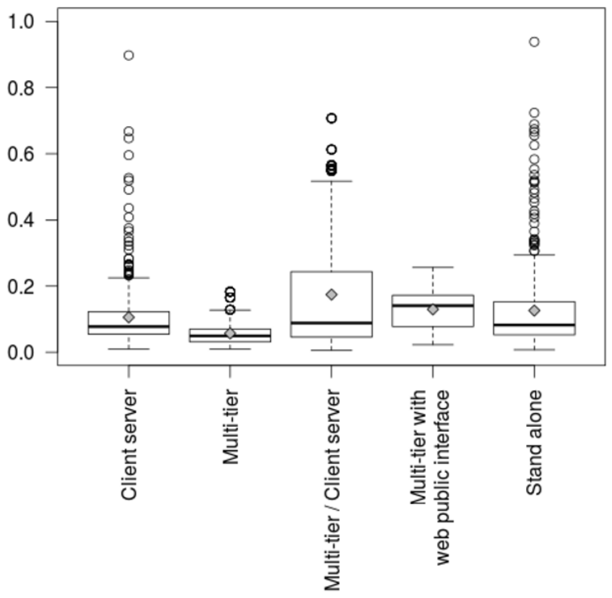

Figure 8. Distributions of enhancement project productivity per architecture type

7. Productivity and economies of scale

The question whether software development exhibits economies (or diseconomies) of scale has been much debated (see Section 9). In general, economies of scale are apparent when it is possible to relate effort and size via models of type $Effort = aSize^b$, with $b < 1$.

In fact, $Effort = aSize^b$ implies that $Productivity = \frac{Size^k}{a}$, where $k = 1 - b$; if $b < 1$, then $k > 0$, and the larger the size, the higher the productivity, as by definition of the economy of scale. On the contrary, if $b > 1$, then $k < 0$, and the larger the size, the smaller the productivity, as in diseconomies of scale. Some studies showed that software development exhibits diseconomies of scale: for instance, this is the case in the well-known COCOMO model [9]. On the contrary, other studies (like [1]) found economies of scale.

To further explore this issue, the existence of $Effort = aSize^b$ models based on ISBSG data was investigated. This type of models is derived by applying the OLS regression after the log-log transformation of data samples. The log-log transformation was used in this research because the data did not comply with the preconditions of OLS about normal distributions.

No statistically significant model could be derived for all new developments, nor for all enhancement projects. Therefore, the economies of scale were studied on data subsets obtained by grouping projects by programming language, business areas, architecture and usage of CASE tools. Grouping project data by these criteria resulted in sufficiently homogeneous datasets, which allowed for the derivation of statistically significant models of effort vs. size.

In the derivation of models, outliers, identified based on Cook's distance, following a consol-

Table 18. Relations between productivities per architecture (new developments)

Architecture	Client server	Multi-tier	Multi-tier/Client server	Multi-tier with web public interf.	Stand alone
Client server		=	<	<	<
Multi-tier	=		<	=	=
Multi-tier/Client server	>	>		>	>
Multi-tier with web public interf.	>	=	<		=
Stand alone	>	=	<	=	

Table 19. Summary data by architecture type for enhancement projects

Architecture	N	Medians Size [UFP]	Medians Effort [PH]	Medians Prod. [UFP/PH]
Client server	443	168	2109	0.078
Multi-tier	45	139	4259	0.049
Multi-tier/Client server	78	339	2860	0.091
Multi-tier with web public interface	51	124	940	0.141
Stand alone	451	175	2096	0.083

idated practice [15] were excluded. The results found are described in the "Outl." column of the tables in the following subsections.

A few statistically significant models featuring quite small adjusted R^2 were found these models are not very interesting, because a small value of R^2 indicates that effort depends mainly on factors other than size and the considered specific characteristics (language, business area, etc.). Accordingly, in the following sections only models featuring adjusted R^2 not less than 0.5 are reported.

7.1. Effect of programming language on economies of scale

By applying the OLS regression after log-log transformation to data samples obtained by grouping new development projects by primary programming language, the models summarized in Table 21 were obtained.

For new development projects that use Java and Visual Basic, the exponent is less than one with 95% confidence: these languages seem to allow for economies of scale. For other languages, it is not possible to decide with 95% confidence if the exponent is less or greater than one, that is, these languages do not cause either economies or diseconomies of scale. It was impossible to obtain statistically significant models only for enhancement projects using PL/1 and ABAP, these are described in Table 22.

PL/I enhancement projects exhibit a diseconomy of scale. Instead, for ABAP enhancement projects no conclusion with 95% confidence could be drawn.

7.2. Effect of business area on economies of scale

By applying OLS regression after log-log transformation to data samples obtained by grouping

Table 20. Relations between productivities per architecture (enhancement projects)

Architecture	Client server	Multi-tier	Multi-tier/Client server	Multi-tier with web public interf.	Stand alone
Client server		$>$	$<$	$<$	$=$
Multi-tier	$<$		$<$	$<$	$<$
Multi-tier/Client server	$>$	$>$		$=$	$=$
Multi-tier with web public interf.	$>$	$>$	$=$		$>$
Stand alone	$=$	$>$	$=$	$<$	

Table 21. Effort models for new development projects grouped by programming languages

Language	Model	Exponent confidence	Adj. R^2	Outl.
C	$7.7\ UFP^{1.032}$	0.837–1.226	0.762	3/40
C++	$15.6\ UFP^{0.962}$	0.657–1.268	0.622	5/31
Java	$37.9\ UFP^{0.769}$	0.656–0.882	0.682	21/107
Oracle	$2.8\ UFP^{1.091}$	0.912–1.271	0.852	8/36
SQL	$12.0\ UFP^{0.931}$	0.677–1.184	0.549	0/45
Visual Basic	$12.8\ UFP^{0.877}$	0.775–0.979	0.714	15/131

Table 22. Effort models for enhancement projects grouped by programming languages

Language	Model	Exponent confidence	Adj. R^2	Outl.
ABAP	$7.8\ UFP^{1.069}$	0.909–1.229	0.827	6/45
PL/I	$5.7\ UFP^{1.190}$	1.028–1.351	0.658	12/123

Table 23. Effort models for new development projects grouped by business area

Business Area	Model	Exponent confidence	Adj. R^2	Outl.
Financial (no Banking)	$8.6\ UFP^{0.955}$	0.672–1.239	0.635	0/28
Telecommunications	$12.3\ UFP^{0.915}$	0.675–1.156	0.563	1/47

new development projects per business area, the models summarized in Table 23 were obtained.

The only two statistically significant models found indicate that both economies or diseconomies of scale may occur. The characteristics of effort models for enhancement projects grouped by business area are given in Table 24.

New developments concerning the financial area (excluding banking) appear to allow for economies of scale.

7.3. Effect of architecture on economies of scale

By grouping new development projects per architecture type it was possible to obtain the models summarized in Table 25.

For new development projects, there is no evidence that architectural types lead to economies or diseconomies of scale. By grouping enhancement projects per architecture type, it was possible to obtain the models summarized in Table 26.

Client server and Stand-alone enhancement projects exhibit economies of scale. Although it is not possible to make statements about multi-tier projects with 95% confidence, stil one can observe that the exponent range is mainly less than one in the 95% confidence range, thus it is likely that economies of scale also exist for multi-tier projects.

7.4. Effect of CASE tools on economies of scale

After grouping projects by the usage of CASE tools, the authors were able to find just one model, concerning enhancement projects with the use of CASE tools. The model is described in Table 27.

No economy or diseconomy of scale is apparent.

8. Threats to validity

Construct validity. The definition of productivity is always a sensitive issue and no universally accepted notion of productivity exists. A fairly widely used notion of productivity was chosen for the research, based on the amount of delivered functionality, quantified via UFP, the most widely used functional size measure. Functional size measures, however, may have some weaknesses [16,17], including: (1) the apparent arbitrariness in the selection of the "complexity" weights used to obtain the value of UFP starting from the Base Functional Components (Internal Logical Files, External Interface Files, External Input, External Outputs, and External Queries); (2) the subjectivity inherent to the counting process; (3) the redundancies of the counted elements. As for (1), the weights are based on an initial study by Albrecht [3]. Although they may need to be updated, they are now a part of the standard definition used by ISO for FP [10, 12]. With reference to (2), the International Function Point Users Group periodically issues new guidelines to reduce the amount of uncertainty in the counting process [4]. Finally, the redundancies may affect the efficiency and cost-effectiveness of measuring and using UFP, but are not a real construct threat. However, UFP somehow (and imperfectly) captures the amount of functionality delivered, unlike such measures as LoC which quantify the amount of code delivered and are not available early in the life cycle, but only after coding, when it is too late to make any useful predictions. Also, just because a measure is objectively quantifiable does not mean that it adequately captures a specific software attribute or is useful in practice.

The main threat with this type of studies is the fact that while there are standard definitions of functional size measures, there is hardly any standard definition of how development (or enhancement) effort should be measured. Therefore, different authors may use differently measured effort data. This may lead to different values for productivity.

Therefore, when considering the comparisons reported in Section 4 the reader should take into account the possible differences in effort measures. For instance, the fact that in Table 12 the found productivity values are all greater than those found by Delorey et al. [8] might be due to different effort measurement criteria. In fact, Delorey et al. [8] collected productivity data by

Table 24. Effort models for enhancement projects grouped by business area

Business Area	Model	Exponent confidence	Adj. R^2	Outl.
After Sales & Services	31.3 $UFP^{0.795}$	0.474 –1.116	0.512	1/26
Financial (no Banking)	110.7 $UFP^{0.540}$	0.35–0.729	0.524	13/44
Inbound Logistics	14.5 $UFP^{0.910}$	0.665–1.155	0.574	5/47

Table 25. Effort models for new development projects grouped by architecture

Architecture	Model	Exponent confidence	Adj. R^2	Outl.
Multi-tier	26.1 $UFP^{0.82}$	0.489–1.155	0.548	2/24
Multi-tier Client server	3.5 $UFP^{1.06}$	0.927–1.188	0.746	24/113
Multi-tier with web public interf.	3.2 $UFP^{1.20}$	0.880–1.523	0.626	11/46

Table 26. Effort models for enhancement projects grouped by architecture

Architecture	Model	Exponent confidence	R^2	Outl.
Client server	30.4 $UFP^{0.82}$	0.752–0.898	0.576	77/443
Multi-tier	49.4 $UFP^{0.84}$	0.618–1.062	0.597	5/45
Stand alone	19.3 $UFP^{0.90}$	0.819–0.987	0.539	65/451

analysing the effort devoted by single programmers to single code changes, while the ISBSG collected data concerning whole projects. At any rate, the relative ranking among the various productivities depending on the programming language according to the study of Delorey et al. and according to this study may still be considered valid.

Finally, an intrinsic limit of the analysis is due to the usage of functional size measures to size software. In fact, these measures do not represent the non-functional parts of requirements. So, developing a project with a relatively small functional requirement but huge non-functional requirements (entailing security, reliability, robustness, portability, etc.) may appear unduly characterized by low productivity.

External validity. The obtained results are based on one of the largest datasets publicly available, with projects coming from many different organizations and countries, so they should be fairly representative of the population of new and enhancements projects.

Even though the ISBSG dataset contains a large number of projects, some skew is possible. For instance, some self-selection phenomenon, e.g. only well-organized projects may report their data to the ISBSG dataset, may not be excluded. However, this is a threat that is hard to eliminate for all datasets that collect data on a voluntary basis.

It is true, however, that a large part of the projects in the ISBSG dataset are representative of consolidated practices and languages, instead of innovative ones. The ISBSG dataset does contain data on projects that are recent and innovative, but not enough to allow for a sensible statistical analysis. However, there is a suspicion that innovative applications will always be in the minority in these datasets, given their recentness. It shouldalso be pointed out that a large number of projects are still carried out with consolidated techniques and languages. For instance, in https://www.tiobe.com/tiobe-index/ the top 50 most popular programming languages are listed, and Java, C and C++ are the top 3.

Internal validity. A possible threat to internal validity may come from the fact that these results are based on projects in which data are collected and later reported to ISBSG. This may not be the case for all projects, but this is a threat for all studies of this kind. To mitigate the possible threat due to the way data are collected and reported to ISBSG, only data of the best two categories were used in the research. Moreover, standard data analysis techniques were used. The

Table 27. Effort models for enhancement projects grouped by the usage of CASE tools

CASE tools used	Model	Exponent confidence	Adj. R^2	Outl.
Yes	17.0 $UFP^{0.94}$	0.843–1.037	0.605	45/285

use of log-log transformations may be a possible threat, because the Least Square Regression is carried out with a different figure of merit than the one it would have without the log-log transformation. However, this transformation was useful because the original data did not comply with the assumptions of the Least Square Regression. Also, log-log transformations are quite common in the Empirical Software Engineering, and specifically in the study of Effort models.

9. Related work

A substantial amount of work was carried out to study the main factors affecting software productivity by proposing and analysing processes, methods, tools, and best practices [18–21]. To the best of the knowledge of the authors, there are three literature reviews on productivity factors in software engineering available in the literature [18, 21, 22]. These works focus on the main dimensions of the product, personnel, project, and process. Each of these dimensions is then characterized by sub-factors: product is related to a specific characterization of software, such as domain, requirements, architecture, code, documentation, interface, size, etc. Personnel factors involve team member capabilities, experience, and motivation. Project factors encompass management aspects, resource constraints, schedule, team communication, staff turnover, etc. Process factors include software methods, tools, customer participation, software lifecycle, and reuse. In this paper, the authors do not focus on a specific dimension, but span their empirical study on the main factors reported in the ISBSG dataset (i.e. primary programming language used to develop each software project, the business area addressed by the project, the architectural type adopted by the project and the use of CASE tools).

Directly referring to the factors analysed in this paper, several studies addresses the relation between programming languages and productivity. For example, in [6,8,23–25] different programming languages are studied to investigate their relation with different code aspects such as program length, programming effort, run-time efficiency, memory consumption, and reliability. In [26], the authors explain productivity in the banking, insurance, manufacturing, wholesale/retail, and public administration sectors, limiting their statistical analysis to 206 business software projects from 26 Finnish companies. In [27], software productivity is studied with a dataset on Chinese software companies. Two research question in this study specifically focus on how the business areas and the primary programming language impact productivity, respectively. As for business areas, low productivity is associated to Telecom and Finance areas, while high productivity is associated to Public Administration, Manufacturing and Energy. In this study, financial projects have high productivity, while manufacturing ones have low productivity. In any case, these results cannot be compared with their outputs since in this study two different datasets were analysed (both for the releases and for geographical locations of the projects). As for the programming language, in [27] it is reported that high level programming languages are found to be more productive (the most productive are ASP, C# and Visual Basic, with a median productivity of 34.68, 18.68, and 9.94 size/effort, respectively).

There has also been a considerable debate regarding economies and diseconomies of scale in software development [9, 28–34]. These studies highlighted that it is quite difficult to determine which factors contribute to producing an overall economy or diseconomy of scale; in fact, different dataset provided different indications. Comstock et al. analysed the ISBSG dataset to derive a model that includes both economies and diseconomies of scale, and can help managers maximize productivity by determining the optimal project size within a particular environment [35]. They considered the same factors as the ones considered in this paper, but with a few important differences: programming languages were considered only in terms of "3rd generation", "4th generation" and "application generators"; moreover, the team size was included in the independent variables of the effort estimation models. This makes the interpretation of the results provided in [35] somewhat problematic as far as (dis)economies of scales are concerned, Productivity is seen there as dependent on size but also on team size, which in it turn is likely to be determined by the size of the program to be developed. As the authors of that work state, "the very presence of Team Size represents a diseconomy of scale: AFP (the size in Function points) relates to the achievement; Team Size relates to the resources consumed" [35]. In fact, the authors conclude that "development exhibits a strong economy of scale with respect to project size, and a similar diseconomy of scale with respect to team size" [35]. This type of finding is consistent with the goals of Comstock et al., but it is of little help for the goals of this study. So, based on the assumption that the team size is chosen to maximize productivity, or to satisfy possible local needs and constraints, the team size is excluded from the independent variables of effort models. In this way, the model of type $Effort = aSize^b$ is obtained for every factor, thus highlighting the role of the considered factor in determining (dis)economies of scale.

10. Conclusion and future work

Software development productivity is an important subject that has often proven to be quite complex to understand and analyse. This paper highlights a few statistically significant results. These results can be considered reliable, since they are based on the analysis of a large public data repository, which is generally considered to be representative of software development practices.

Specifically, it was found out that the primary programming language had a significant effect on productivity of new development projects. On the contrary, the productivity of enhancement projects appears much less dependent on programming languages. The business area and the architecture have a significant effect on productivity of both new development and enhancement projects. No evidence of the impact of the use of CASE tools on productivity was found, for either new developments or enhancement projects.

In addition, it was found that the productivity of new development projects tends to be higher than that of enhancement projects. Also, the results of our analyses show productivity values obtained that are higher, for each programming language, than those of the reference works on the subject, carried out by Jones, and for open-source software, as reported by Delorey et al.

It was also analysed what factors seem to have an impact on the presence of economies and diseconomies of scale. For instance, economies of scale for new development projects using Java or Visual Basic were found and also diseconomies of scale for enhancement projects concerning applications written in PL/1, while neither economies or diseconomies of scale could be found for other projects. Economies of scale were also found for enhancement projects in the financial area (excluding banking), and for enhancement projects concerning application featuring stand alone or client server architectures.

Future work will focus on:

– investigating whether other factors may influence productivity and the existence of economies or diseconomies of scale;
– carrying out analysis on additional datasets;
– using different measures of productivity, for instance, based on different functional size measures.

Acknowledgments

This work has been partially supported by the "Fondo di ricerca d'Ateneo" funded by the Università degli Studi dell'Insubria.

References

[1] R. Premraj, M. Shepperd, B. Kitchenham, and P. Forselius, "An empirical analysis of software productivity over time," in *Proceedings of the 11th IEEE International Software Metrics Symposium*, ser. METRICS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 37–37.

[2] ISBSG, "International Software Benchmarking Standards Group – Worldwide software development: The benchmark, release 12," 2015.

[3] A. Albrecht, "Measuring application development productivity," in *Joint SHARE/GUIDE/IBM Application Development Symposium.* IBM, 1979.

[4] Function Point counting practices manual – release 4.2, International Function Point Users Group, (2004)

[5] L. Lavazza, S. Morasca, and D. Tosi, "An empirical study on the effect of programming languages on productivity," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, ser. SAC '16. New York, NY, USA: ACM, 2016, pp. 1434–1439.

[6] C. Jones, Programming languages table. Release 8.2, Software Productivity Research, Inc., (1996). [Online]. https://engenhariasoftware.files.wordpress.com/2008/06/conversao.pdf

[7] C. Jones, "Function points as a universal software metric," *SIGSOFT Software Engineering Notes*, Vol. 38, No. 4, Jul. 2013, pp. 1–27.

[8] D.P. Delorey, C.D. Knutson, and S. Chun, "Do programming languages affect productivity? A case study using data from open source projects," in *Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development*, ser. FLOSS '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 8–8.

[9] B.W. Boehm, *Software Engineering Economics*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1981.

[10] *Software Engineering NESMA Functional Size Measurement Method, Version 2.1, Definitions and counting guidelines for the application of Function Point Analysis, International Organization for Standardization*, ISO Std. ISO/IEC 24 750:2005, 2005.

[11] ISBSG, "The performance of real-time, business application and component software projects," The Common Software Measurement International Consortium & The International Software Benchmarking Standards Group, Tech. Rep., Apr. 2011.

[12] *Software engineering – IFPUG 4.1. Unadjusted functional size measurement method – Counting Practices Manual*, ISO Std. ISO/IEC 20 926:2003, 2003.

[13] W.H. Kruskal and W.A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American Statistical Association*, Vol. 47, No. 260, 1952, pp. 583–621. [Online]. http://www.jstor.org/stable/2280779

[14] B.A. Myers, J.F. Pane, and A. Ko, "Natural programming languages and environments," *Commun. ACM*, Vol. 47, No. 9, Sep. 2004, pp. 47–52.

[15] L. Lavazza and S. Morasca, "Software effort estimation with a generalized robust linear regression technique," in *16th International Conference on Evaluation Assessment in Software Engineering (EASE 2012)*, May 2012, pp. 206–215.

[16] B. Kitchenham, "The problem with function points," *IEEE Software*, Vol. 14, No. 2, Mar. 1997, pp. 29–31.

[17] B. Kitchenham, S.L. Pfleeger, and N. Fenton, "Towards a framework for software measurement validation," *IEEE Transactions on Software Engineering*, Vol. 21, No. 12, Dec. 1995, pp. 929–944.

[18] B.W. Boehm, "Improving software productivity," *Computer*, Vol. 20, No. 9, Sep. 1987, pp. 43–57.

[19] A. Trendowicz and J. Munch, "Factors influencing software development productivity – state-of-the-art and industrial experiences," *Advances in Computers*, Vol. 77, 2009, pp. 185–241.

[20] J. Vosburgh, B. Curtis, R. Wolverton, B. Albert, H. Malec, S. Hoben, and Y. Liu, "Productivity factors and programming environments," in *Proceedings of the 7th International Conference on Software Engineering*, ser. ICSE '84. Piscataway, NJ, USA: IEEE Press, 1984, pp. 143–152. [Online]. http://dl.acm.org/citation.cfm?id=800054.801963

[21] K.D. Maxwell, L. Van Wassenhove, and S. Dutta, "Software development productivity of european space, military, and industrial applications," *IEEE Transactions on Software Engineering*, Vol. 22, No. 10, Oct. 1996, pp. 706–718.

[22] S. Wagner and M. Ruhe, "A structured review of productivity factors in software development," Institut für Informatik, Technische Universität München, techreport TUMI0832, 2008.

[23] L. Prechelt, "An empirical comparison of seven programming languages," *Computer*, Vol. 33, No. 10, 2000, pp. 23–29.

[24] K. Kennedy, C. Koelbel, and R. Schreiber, "Defining and measuring the productivity of programming languages," *The International Jour-*

nal of High Performance Computing Applications, Vol. 18, No. 4, Nov. 2004, pp. 441–448.

[25] R. Klepper and D. Bock, "Third and fourth generation language productivity differences," *Communications of the ACM*, Vol. 38, No. 9, Sep. 1995, pp. 69–79.

[26] K.D. Maxwell and P. Forselius, "Benchmarking software-development productivity," *IEEE Software*, Vol. 17, No. 1, Jan. 2000, pp. 80–88.

[27] M. He, M. Li, Q. Wang, Y. Yang, and K. Ye, "An investigation of software development productivity in China," in *International Conference on Software Process*. Springer, 2008, pp. 381–394.

[28] D.L. Nazareth and M.A. Rothenberger, "Assessing the cost-effectiveness of software reuse: A model for planned reuse," *Journal of Systems and Software*, Vol. 73, No. 2, 2004, pp. 245–255.

[29] R.D. Banker and C.F. Kemerer, "Scale economies in new software development," *IEEE Transactions on Software Engineering*, Vol. 15, No. 10, Oct. 1989, pp. 1199–1205.

[30] J.E. Matson, B.E. Barrett, and J.M. Mellichamp, "Software development cost estimation using function points," *IEEE Transactions on Software Engineering*, Vol. 20, No. 4, Apr. 1994, pp. 275–287.

[31] B.A. Kitchenham, "The question of scale economies in software-why cannot researchers agree?" *Information and Software Technology*, Vol. 44, No. 1, Jan. 2002, pp. 13–24.

[32] R.D. Banker, H. Chang, and C.F. Kemerer, "Evidence on economies of scale in software development," *Information and Software Technology*, Vol. 36, No. 5, 1994, pp. 275–282.

[33] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," *IEEE Transactions on Software Engineering*, Vol. 30, No. 12, Dec. 2004, pp. 1023–1035.

[34] F.P. Brooks, Jr., *The Mythical Man-month (Anniversary Ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[35] C. Comstock, Z. Jiang, and J. Davies, "Economies and diseconomies of scale in software development," *Journal of Software Maintenance and Evolution*, Vol. 23, No. 8, Dec. 2011, pp. 533–548.