Empirical Findings on Team Size and Productivity in Software Development

D. Rodríguez[a,b,*], M.A. Sicilia[a], E. García[a], R. Harrison[b]

[a]*Department of Computer Science, University of Alcalá, Ctra. Barcelona, Km. 31.6, 28871 Alcalá de Henares, Madrid, Spain*
[b]*School of Technology, Oxford Brookes University Wheatley Campus, Oxford OX33 1HX, UK*

Abstract

The size of software project teams has been considered to be a driver of project productivity. Although there is a large literature on this, new publicly available software repositories allow us to empirically perform further research. In this paper we analyse the relationships between productivity, team size and other project variables using the International Software Benchmarking Standards Group (ISBSG) repository. To do so, we apply statistical and machine learning approaches to a preprocessed subset of the ISBSG repository to facilitate the study. The results show some expected correlations between productivity, effort and time as well as corroborating some other beliefs concerning team size and productivity. In addition, this study concludes that in order to apply statistical or data mining techniques to these type of repositories extensive preprocessing of the data needs to be performed due to ambiguities, wrongly recorded values, missing values, unbalanced datasets, etc. Such preprocessing is a difficult and error prone activity that would need further guidance and information that is not always provided in the repository.

Keywords:
Team size, productivity, effort estimation datasets, ISBSG repository

*Corresponding author. This work was carried out while visiting Oxford Brookes University.

Email addresses: daniel.rodriguezg@uah.es (D. Rodríguez), msicilia@uah.es (M.A. Sicilia), elena.garciab@uah.es (E. García), rachel.harrison@brookes.ac.uk (R. Harrison)

1. Introduction

Among the important factors that affect productivity, project team size has been considered a key driver. The IEEE Std. 1045-1992 [20] defines the *productivity ratio* as the relationship of an output primitive and its corresponding input primitive, where the input primitive refers to the effort (staff-hours) to develop software products and the output primitive refers to either source statements, function points or documents. Productivity is sometimes also known as efficiency [39].

Among the many managerial decisions that need to be made in a software project, personnel related factors are among the ones affecting productivity most [38]. This raises the concern on finding empirical evidence about the relationships between project attributes, productivity and staffing levels that can help optimise managerial decisions. Concretely, it is commonly acknowledged that the time spent in communication among team members increases with the size of the team. Project team size therefore affects schedule decisions, which are also acknowledged as an important factor in project success [40]. Furthermore, team size is important when making decisions about the structure of teams and the eventual partition of projects into smaller subprojects. If an optimal team size could be found, then the decomposition of projects into smaller pieces would become a key management practice with direct implications in the decision of distributing project teams.

Team size is also considered one of the more influential factors in software productivity by the ISBSG organization: *"The ISBSG data shows that there are three main factors that impact software development productivity: programming language, development platform and team size. The first two have the most significant effect but it is also important to consider the impact of team size. The latest ISBSG Special Report reveals that teams of nine or more are significantly less productive than smaller teams"* [22]. The rule of thumb for the team size *less than nine*, provides an interesting management in-sight. However, there is a need for additional research into data and models, and it should be complemented by more elaborate models that help staffing decisions. This rule of thumb is obviously directed to straightforward decision making, but it is important to contrast such rules with the available evidence, and eventually, to develop methods for adapting them to the specificities of each particular organization.

This paper aims to provide a systematic empirical study of the impact of team size and other development factors in the productivity of software projects by using statistical analyses after preprocessing an ISBSG repository (release 10). A better understanding of the interactions between such factors could help in both defining estimation models as well as decision making processes for project managers. There is also a need to study publicly available software engineering repositories to empirically validate their usefulness.

The rest of this paper is structured as follows. Section 2 covers related work on team size and productivity in software development. Section 3 describes the ISBSG repository and the necessary preprocessing for the its analysis. Then, Section 4 provides an exploratory study of the significance of team size and productivity according to the data in the ISBSG repository with statistical and data mining approaches. Finally, conclusions and future work are provided in Section 6.

2. Related Work

According to the Project Management Institute (PMI)[1], Software project management mainly consists of applying knowledge, skills, tools and techniques to project activities in order to meet or exceed stakeholder needs and expectations from a project [34]. To do so, project managers need to define the project tasks, their duration and dependencies and assigning resources to them.

The influence of team size in the productivity of software teams has been a topic of research in software engineering for many years. In his well-known book, Brooks [8] claimed in 1975 that assigning more programmers to a project running behind schedule will make it even later, due to the time required for the new programmers to learn about the project, as well as the increased communication overhead. Although Brooks also stated that this law is outrageously oversimplified, he provided some facts about why estimates are not accurate: (i) estimating techniques are poorly developed and we are optimistic by nature; (ii) estimating techniques fallaciously confuse effort with progress, hiding the assumption that people and months are interchangeable; (iii) inherent uncertainty of estimates; (iv) schedule progress is poorly monitored; and (v) when there is a schedule slippage, software managers tend to increase manpower, making things worst, which is known as

[1] `http://www.pmi.org/`

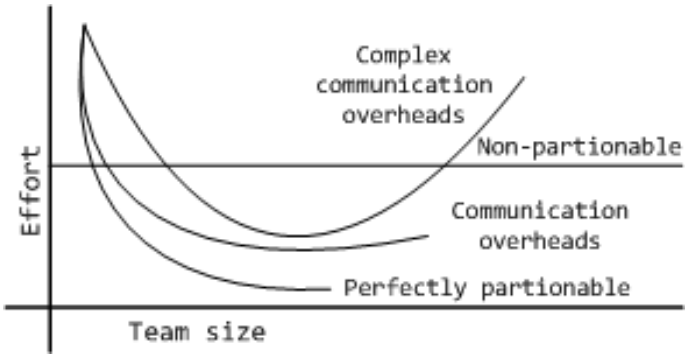

Figure 1: Brooks' Tasks Classification following (following [13])

Brooks' law. Brooks also classifies tasks as (i) perfectly partitionable task; (ii) unpartitionable task, (iii) partitionable task requiring communication; and (iv) task with complex interrelationships. His view is that most tasks in software engineering belong the the last category, task with complex interrelationships. These relationships between effort and team size are summarised in Figure 1.

Abdel-Hamid and Madnick developed a System Dynamics model [2, 1] to study such relationships. Among other things, their model was used to analyse Brooks' law by applying different staffing policies on cost and schedule in a specific project, the NASA DE-A project. The authors conclude that adding more people to a late project always causes it to become more costly but does not always cause it to finish later.

Phister [27] provided one of the first models that accounted for team size as a critical variable related to productivity. Smith et al. [37] analysed the impact on software development effort of the following factors: (i) team size defined as number of people working on a module; (ii) concurrency defined the degree to which people work together or dependently in a module; (iii) intensity, which measures the degree of schedule compression; and (iv) fragmentation, which examines the degree to which a team's time is fragmented over multiple modules. The authors modified several versions of the COCOMO model and also created a new parsimonious model considering only previously defined factors. The authors concluded that the parsimonious model was superior to the original and modified COCOMO models while using fewer factors. Regarding the team size factor, the authors also concluded that team size does not significantly affect effort in the studied development

environment.

Trendowicz and Münch [38] report on a comprehensive study of factors affecting productivity through a systematic literature review [23] and industrial experiences consisting of industrial projects, workshops and surveys. The authors categorise multiple factors into the following groups: product, personnel, project and process factors. The authors also analyse their reported frequency in four contexts: cost modeling, productivity measurement, project data repositories and studies on software process improvement. The authors consider context factors as those applicable to a variety of environments used to build models. The factors most frequently cited include the programming language, domain and development type. Also, influence factors are those included in the model to explain the variability of context factors. In this case, some of the most influential factors include: team capabilities and experience, software complexity, project constraints and tool usage. According to the authors, the human side of the software development process is the most important one followed by tools and methods and contrary to intuition and belief, reuse is not a key factor in productivity.

One important point is that staffing is considered to vary during the software development life cycle and it can be adjusted by the Rayleigh distribution as first described by Norden [30, 29]. Putman [35] extended Norden's model into the Software LIfe Cycle Management (SLIM) model in what he defined the *software equation* based on the hypothesis that the size of software project is proportional to productivity, effort and time. According to Putnam in QSM[2], there is an optimum team size, defined as the one which allows the development team to achieve the maximum productivity with shortest schedule and lowest cost without affecting the final outcome. Putman also states that the optimum team size is dependent on a number of variables including: (i) the size of code to be developed and reused, (ii) project complexity and (iii) the degree to which schedule or cost is the overriding schedule constraint. Following a quantitative approach, Putman presents a study that analyses the productivity of projects stratified into five groups (1.5-3 staff, 3-5 staff, 5-7 staff, 9-11 staff, 15-20 staff) from a sample of 491 projects. Putman concluded that productivity is higher for smaller teams with an optimum team size of 3 to 5 staff but with very similar values for teams between 5 to 7 staff. In relation to the schedule, Putman stated

[2] `http://www.qsm.com/process_01.html`

that schedule performance improves up to a team size of 9 to 11 people but with larger teams the schedule performance decreases. To sum up, teams of 9 or more people represent more effort and cost with exponential growth after such a threshold. Pillai and Sukumaran Nair [33] proposed a further refined Putman's model for the dynamics of manpower to provide better adjustment capabilities. Other statistical models for estimation in general and productivity in particular include the work by Sentas et al. [36] which report on the use of Ordinal Regression to estimate the productivity divided into four intervals.

More recently, this problem has been approached using Search Based Software Engineering (SBSE) techniques which mainly consist of the application of metaheuristic techniques in Software Engineering [18]. The application of SSBSE to project management has generated a research area also known as Search-based Software Project Planning (SBSPP). For example, Di Penta et al. [13] analysed Brooks' law using search based techniques (following a previous work [6]);for perfectly partitionable tasks, we have *time=effort/persons*. However as communication effort needs to be taken into account, we have *time=(effort/persons)·commEff*, where *commEff* is a measure of communication effort. Simulations confirmed that (i) different communication overheads affect statistically the duration time, (ii) increases in communications overheads favour a larger number of smaller teams in opposition to a smaller number of larger teams. Di Penta et al. [12, 7] have also compared different approaches (Genetic Algorithms, Stochastic Hill Climbing and Simulated Annealing) to SBSPP. Previously the authors focused on exploring the staffing problem and communication overheads (Brook's law) [32] using genetic algorithms. Other authors such as Alba and Chicano [3, 4] have also applied genetic algorithms as a technique to optimise the assignation of people to software development tasks. More recently, multi-objective approaches are being applied for example by Gueorguiev et al. [17] and also by Chicano et al. [9]. Kremmel et al. [25, 26] have also tackled allocation of resources to projects with metaheuristics. In our paper, we focus on studying some important attributes rather than suggesting techniques to allocate resources. The clarification of important attributes and thresholds can help to define constraints that can be used with the search techniques.

The problem of productivity is related to the problem of economies and dis-economies of scale in Software Engineering. For example, Dolado [14] conducted exhaustive research of cost estimation models using regression and genetic programming with most of the publicly available repositories

available at the time. He concluded that there is no clear economy or diseconomy of scale in software development. Comstock [10] has investigated the same problem of economy and diseconomy of scale using the ISBSG repository. The equations generated using three models (their own, Putman and COCOMO) showed an economy of scale for size and diseconomy of scale for team size. A number of authors have also analysed different releases of the ISBSG repository in relation to effort estimation. For example, Pendharkar and Rodger [31] studied the impact of team size on development effort. According to the authors, their study validates Brooks' law and other interesting results include that there is no relationship between software development effort and using CASE (Computer Assisted Software Engineering) tools or the type of programming language (3GL or 4GL) and development effort. Haričko et al. [19] explored the relationship between project size, effort and team size and proposed an approach based on the Generalised Reduced Gradient and Sequential Quadratic Programming to define the optimal number of developers to minimise the development effort using data from the ISBSG repository. In this paper we also analyse the ISBSG repository which is probably the largest available repository in both the number of projects and attributes.

3. The ISBSG Repository

The International Software Benchmarking Standards Group (ISBSG)[3], a non-profit organization, maintains a software project management repository from a variety of organizations. The ISBSG checks the validity and provides benchmarking information to companies submitting data to the repository. Projects are defined using over 60 attributes such us number of function points and type of function points (IFPUG, COSMIC, Mark II, etc.), team size, software lifecycle phases, etc. The ISBSG attributes can be classified as follows:

- Project context such as type of organization, business area, and type of development.
- Product characteristics such as application type user base.

[3] `http://www.isbsg.org/`

- Development characteristics such as development platform, languages, tools, etc.
- Project size data: different types of function points (IFPUG, COSMIC, etc.)
- Qualitative factors such as experience, use of methodologies, etc.

However, before applying statistical or machine learning approaches to the dataset, there are a number of issues to be taken into consideration regarding data preparation. We first needed to preprocess data as explained in the next section.

3.1. Preparation and Validation of the ISBSG Repository

This section describes the preprocessing or data cleaning process carried out across different ISBSG releases. To do so, we performed instance (project) selection and attribute selection.

Firstly, it was necessary to remove projects that have no significance to the analysis due to poor quality or lack of data, inspecting instances for each attribute as follows:

- *Rating of the data.* The ISBSG uses two attributes, *Data Quality Rating* and *Unadjusted Function Point Rating* as a measure of reliability for each project.
  - Data Quality Rating. Projects are classified from A (where the submission to the ISBSG consortium satisfied all criteria for seemingly sound data) to D (where the data had some fundamental shortcomings). According to ISBSG, only projects classified as A or B should be used for statistical analysis. Therefore, all project instances labelled as C or D were removed.
  - Unadjusted Function Point Rating. This attribute refers to the quality of the functional size. As above, only projects classified as A or B should be used for statistical analysis.
- *Count Approach.* A description of the technique used to size the project. The projects that used other size estimating method (*Derived count approach*) than IFPUG[4], NESMA[5], Albrecht or Dreger were removed,

[4]International Function Point Users Group: `http://www.ifpug.org/`
[5]Netherlands Software Metrics Users Association: `http://www.nesma.nl/`

since they represent small portions of the database. NESMA is a compliant variant of IFPUG and the differences between them are considered to have a negligible impact on the results of function point counts [28]. Counts based on Albrecht's [5] technique were not removed since in fact IFPUG is a revision of these techniques, similarly, the Dreger method refers to the book [15], which is simply a guide to IFPUG counts. All these variants were included.

- The *Recording Method* attribute represents the method used to obtain work effort data. Projects with *Recording method* for total effort other than *Staff hours (Recorded)* were removed. This method reports the work effort from a daily record of all the work effort expended by each person on project related tasks. Others values such as *Staff hours(Derived)*, or *Productive Time Only* are not reliable because they are considered to be more subjective.

- The *Resource Level* Data is collected about type of effort included in the work effort data reported. Out of the four possible levels, only *Level 1* was considered, i.e., development team effort (e.g., project team, project management, project administration). Development team support, computer operations involvement, and end users or clients, are not included as we are only interested in the project construction effort.

- Next we considered all nominal attributes. It was necessary to perform a homogenization process, for example, many entries are blank while others are labeled as "*don't know*". Another example is that the same programming language is described using equivalent but different labels such as COBOL 2 or COBOL II. It is, therefore, necessary to unify them with the same value. A similar process is performed for all attributes. We also tabulated all nominal attributes to obtain their frequencies. Nominal values with less than 1% out of the total number of projects were removed as they were considered not representative. For example, the *Language Type* attribute can be 2GL, 3GL, 4GL, 5GL, and ApG but the number of projects developed with 2GL, 5GL and ApG languages was less than 1%; these projects were removed, leaving only 3GL and 4GL.

Secondly, we performed attribute selection. The entire ISBSG repositories have more than 60 attributes grouped around data rating, sizing, effort,

productivity, schedule, quality, grouping attributes, architecture, documents and techniques, project attributes, product attributes, effort attributes, size attributes and size other than functional size measures. After cleaning and removing instances (projects) as described, some attributes contained only one value. Attributes with a single value do not provide any usable information for data mining or statistical analyses. Even then, it was not be possible to consider all remaining attributes in our analysis. Attributes that can confuse the statistical or machine learning algorithms were further removed. For example attributes in which the percentage of missing values is larger than 85% were removed.

Several attributes represent different way of measuring the same project characteristic. For example, among all productivity attributes, we selected the *Normalised Level 1 Productivity Delivery Rate* as it is the development productivity recommended by the ISBSG, and as a consequence, the *Normalised Level 1 Work Effort* was also selected for the effort. Other productivity or effort attributes were removed.

Table 1 shows a summary of their types, range and percentage of missing values of the following attributes selected with approximately one third of all instances:

1. Functional Size (FP). This is the count of unadjusted Function Points (FP) [5, 21] using IFPUG or its variants.
2. Normalised Level 1 Work Effort (Eff). Development team only effort for the full development life-cycle (this can be an estimate when the project did not cover the full life-cycle). Effort is measured in staff hours.
3. Normalised Level 1 Productivity Delivery Rate (PDR). Number of hours per functional size unit. This is calculated from *Normalised Level 1 Work Effort* for the development team only divided by *Functional Size* (Unadjusted Function Points). Therefore, smaller values represent better productivity.
4. Development Type ($DevType$). Whether the development was a *new development*, *enhancement* or *re-development*.
5. Organization type ($OrgType$). Type of organization that submitted the project (e.g.: Banking, Manufacturing, Retail).
6. Development Platform ($Platf$). It defines the primary development platform, (as determined by the operating system used). Each project

Table 1: Relevant ISBSG Attributes, their Types and % of Missing Values (after preprocessing)

Attribute	*Type (Range)*	*% Missing*
FP	Ratio [3-4,911]	0%
Eff	Ratio [4-73,920]	0%
PDR	Ratio [0.1-378.1]	0%
DevType	Nominal {Enhancement, NewDev}	0%
OrgType	Nominal {Insurance, Ordering, VoiceProvisioning, Banking, Manufacturing, Communications, FinPropBusServ}	40%
Platf	Nominal {MF, PC, Multi, MR}	51%
LangType	Nominal {3GL, 4GL}	25%
PrimaryProgLang	Nominal {Java, C, ORACLE, COBOL, PL, VB, SQL, C++, NATURAL}	25%
MTS	Ratio [0.5-309]	58%
ATS	Ratio [1-41]	79%

is classified as: PC, MR (Mid-Range), MF (Mainframe) or $Multi$ (Multi-platform).

7. Language Type (*LangType*). Language type used for the project, $3GL$, $4GL$.
8. Primary Programming Language (*PrimaryProgLang*). The primary language used for the development: Java, C++, PL/1, Natural, COBOL, etc.
9. Maximum Team Size (MTS). The maximum number of people that worked at any time on the project (for the Development Team).
10. Average Team Size (ATS). The average number of people that worked on the project (calculated from the team sizes per phase).

Table 2 shows the descriptive statistics for the numerical attributes when all 951 project instances of ISBSG release 10 are taken into account.

We checked the standard skewness and kurtosis values together with the histograms (Figures 2 to 4) and found that none of the relevant variables follows the normal distribution. Although this is quite intuitive from the histograms, it was also confirmed with the Shapiro-Wilk test for normality. Therefore we cannot apply parametric tests with this data.

Table 2: Descriptive Statistics for the Selected Continuous Variables in the ISBSG after Preprocessing

	FP	Eff	PDR	MTS	ATS
Count	951	951	951	401	200
Avg	302.7	3375.3	17.9	8.4	7.9
Med	135	1521	11.3	5	5
Std Dev	490.1	6055.3	26.9	17.0	7.2
Min	3	4	0.1	0.5	1
Max	4911	73920	387.1	309	41
Rng	4908	73916	387	308.5	40

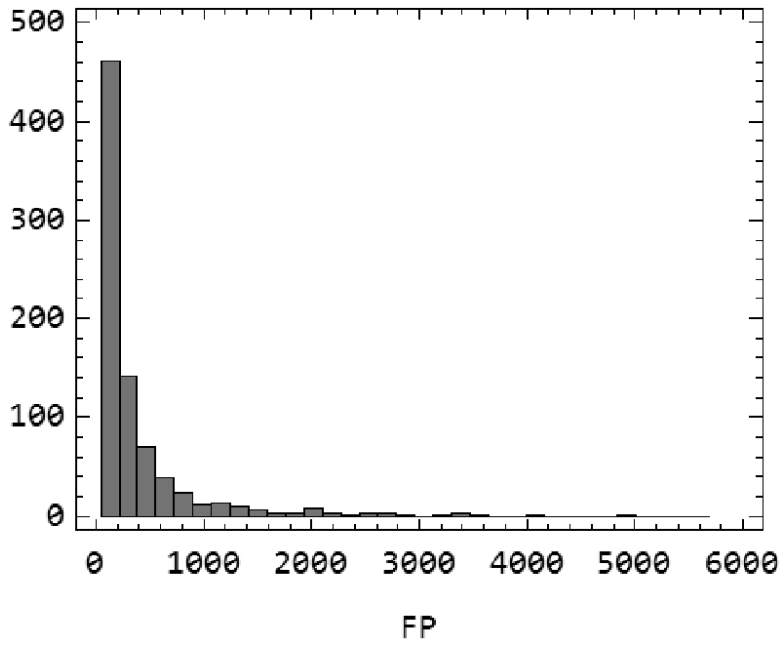

Figure 2: Histogram for the Functional Size (FP) Variable

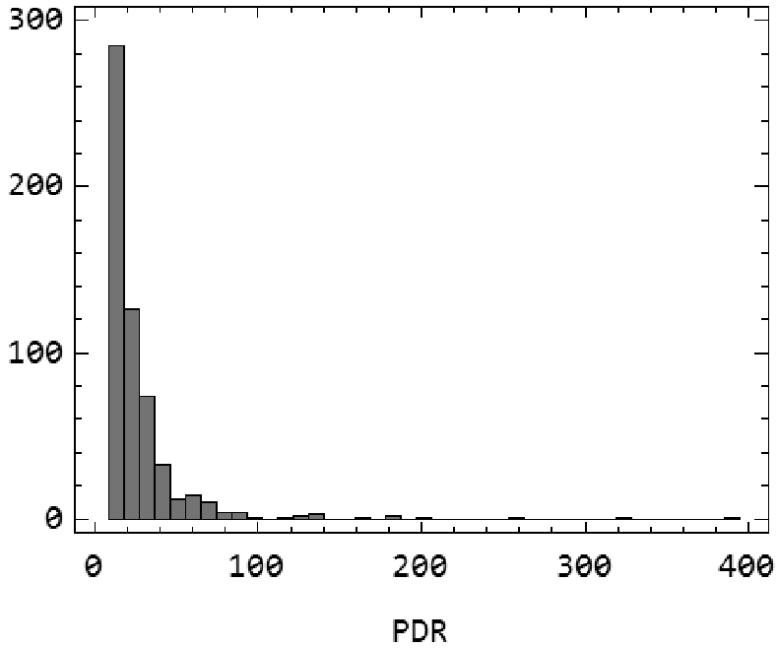

Figure 3: Histogram for the Productivity (PDR) Variable

4. Analysis of the Productivity and Team Size in the ISBSG Repository

In this section, we analyse relationship between between team size and productivity together with other attributes. Exploratory analyses were performed on releases 8, 9 and 10 of the ISBSG database after preprocessing the data following the same procedure for each release as described previously. However, all analyses described here were performed using the release 10 as each release of ISBSG mainly increases the number of reported projects.

As a first analysis, Table 3 shows the correlations of those variables used later on using the non-parametric Spearman's rank method as measure of statistical dependence between two variables. There is an obvious correlation between the MTS and the ATS but there is a degree of association between the rest of the variables, especially between normalized work effort and the team size measures.

When considering only those projects reporting both the team size attributes (MTS and ATS) after the preprocessing and after, the descriptive statistics regarding team size and productivity are shown in Table 4.

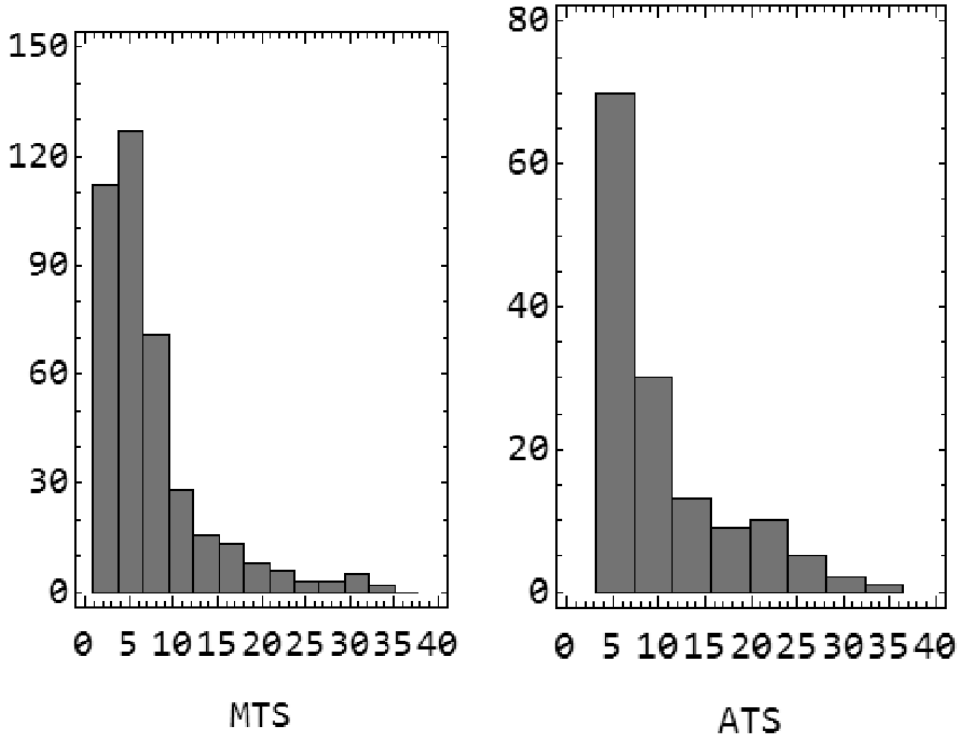

Figure 4: Histogram for the Maximum Team Size (MTS) and Average Team Size (ATS) Variables

Table 3: Spearman's Rank Correlations between Relevant Numerical Variables

	FP	Eff	PDR	MTS	ATS
FP	—	0.67	-0.32	0.41	0.20
Eff		—	0.43	0.76	0.73
PDR			—	0.42	0.57
MTS				—	0.90

Table 4: Descriptive Statistics for Projects Reporting Both ATS and MTS (after Preprocessing)

ISBSG	*Avg/StdDev ATS*	*Avg/StdDev MTS*	*3Q ATS*	*3Q MTS*	*Avg/StdDev PDR*
R10 ($n = 200$)	7.9 / 7.21	9.15 / 8.79	10.0	11.0	15.94 / 17.21

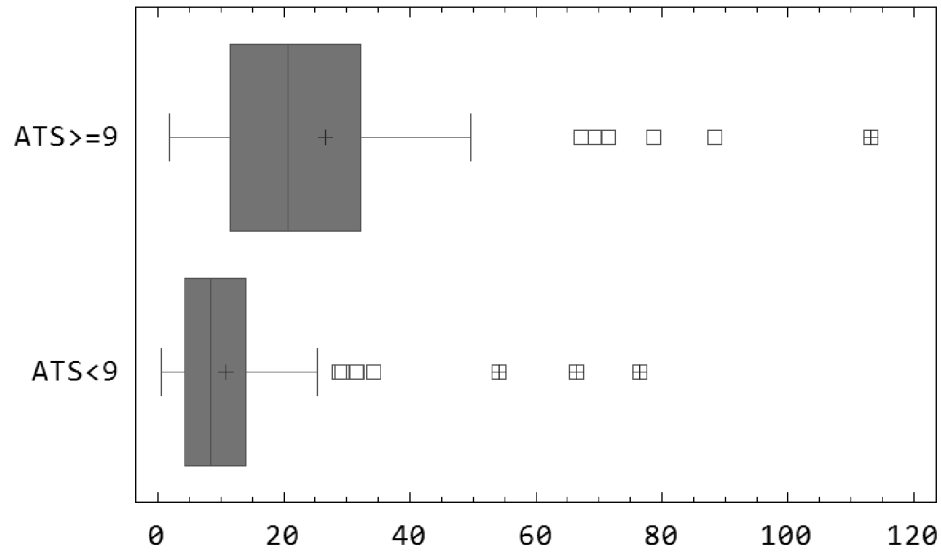

Figure 5: Box Plot for $ATS < 9$ and $ATS \geq 9$.

Table 5: Statistics for Variations between MTS and ATS in R10

	Count	*MTS Mean/Std Dev*	*FP Mean/Std Dev*	*PDR Mean/Std Dev*
$(MTS - ATS) > 0$	46	9.05/11.99	544.30/579.24	13.98/20.72
$(MTS - ATS) = 0$	153	9.09/7.63	285.73/369.12	16.62/16.05

An important initial observation is that 75% of the data is concerned with a team size of less than 10 people. This suggests that predictive models should consider at least a segmentation of the database into two parts. As expected from the literature, productivity is worst for those projects with ATS larger or equal to 9 people (the value suggested by the ISBSG repository [22]). Comparing productivity medians instead of the means of as the data is very asymmetric and skewed, larger values (less productivity) correspond to larger teams (see Figure 5). Also the Wilcoxon Rank Sum test indicates that this difference is statistically significant between the two distributions at the 95.0% confidence level ($p - value = 2.2e^{-16}$).

When we compare productivity between those projects reporting a difference between MTS and ATS, there is a large variability in their average and standard variation as can be seen in Table 6. According to the Wilcoxon test, there is a statistically significant difference between these two groups at the 95.0% confidence level ($p - value = 0.0081$). However, on a closer look at these two groups, we observe that there are important differences in the de-

Table 6: Descriptive Statistics for R10

	$PDR\ (diff = 0)$	$PDR\ (diff \neq 0)$
$Count$	152	47
$Average$	16.3	13.7
$Median$	10.9	7.1
$StdDev$	15.7	20.6
Min	0.5	0.8
Max	88.3	113.2
$Range$	87.8	112.4
$3Q$	20.7	16.1

velopment, organization and business types of organizations. First, although there are missing values, all projects that reported that there is no difference between MTS and AVT belong to the telecommunications sector. Second, those projects not reporting any difference are mainly enhancement projects (120 out of 152) and all of them belong to the telecommunications business type. In contrast, those reporting differences are mainly new developments (33 out of 47) and 15 out of 47 belong to banking organisations. The other domains are blanks. This also affects the type of languages (mainly COBOL in the banking group). A risk from analysing these values is that both groups seem completely disjoint. It is possible that a single organisation reported these projects providing with both team size values so we could possibly comparing the productivity of a single organisation against the rest (this is unknown from the data). Therefore, further analyses of the productivity values in different organization types, programming languages and development types were performed taking into account all projects and the ATS.

A factor that differentiates productivity is whether the projects are new developments or enhancement projects. Table 7 shows the descriptive statistics for productivity between new developments and enhancements projects. The test to compare both distributions shows that there is a statistically significant difference between then at the 95% level ($p - value = 1.948e^{-07}$). It is difficult to generalise due to the unknown origin of the data, but it is important from the manager's point of view to recognise that enhancement project will be more productive, probably due to the experience of personnel in both the domain of the application and the application itself.

As stated previously, according the ISBSG there are three main factors that affect productivity: (i) programming language, (ii) development platform and (iii) team size. In order to analyse how these variables affect pro-

Table 7: Productivity Statistics for New Developments vs. Enhancement Projects

	PDR (New Dev)	*PDR (Enh)*
Count	733	219
Average	19.57	12.43
Median	12.4	7.6
StdDev	29.46	14.21
Min	0.3	0.1
Max	386.8	113.2
Range	387.1	113.1
3Q	22.3	15.8

ductivity using one-way Anova, Naperian logarithmic transformation was applied to continuous variables in order to approximate them to the normal distribution. Figure 6 shows the normal probability plot for the productivity variable (PDR). The values along the straight line show that we can assume normality of the transformed function.

As expected, there are large differences in productivity across the different languages which were analysed using one-way Anova to compare the means of productivity values. Such differences are statistically different according to $p - value$ of the $F - test$ which is practically 0 at 95% confidence level.

Table 8 shows the mean productivity for the programming languages selected in the analysis, the standard error of each mean and its interval based on Fisher's Least Significant Difference (LSD) procedure (the intervals are constructed in such a way that if two means are the same, their intervals will overlap 95% of the time). Figure 7 shows the box plots of the means of the transformed productivity variable by programming language. Table 8 also shows the multiple range tests of previous table intervals to determine which means are significantly different from which others. It is worth noting that there is a large number of different languages reported in the primary programming language attribute but after the preprocessing those listed here are the ones which have a reasonable number of instances for statistical analysis. Also, many instances report 3GL or 4GL as a primary language (96 and 34 respectively); this information is also reported in the *language type* attribute) and those projects not considered in this analysis.

Among the 9 languages considered, Visual Basic (VB) was the most productive which seems quite reasonable considering its visual component and simplicity when compared with the rest of the languages considered. There is a large variability among the mainframe related languages (Natural, PL/1,

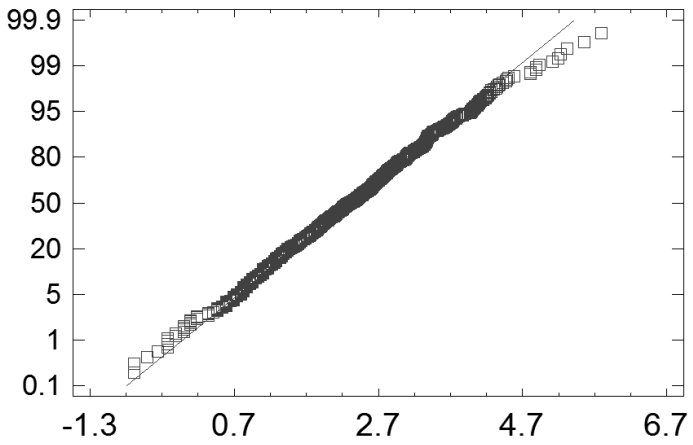

Figure 6: Normal Probability Plot for the Productivity Variable

Table 8: Means for $ln(PRD)$ by Programming Language with 95% LSD intervals and Homogeneous Groups

	Count	*Mean*	*Std Err*	*Lower limit*	*Upper limit*	*Homog. Groups*
VB	74	1.98	0.11	1.83	2.14	A
$NATURAL$	16	2.21	0.24	1.87	2.55	ABC
$ORACLE$	82	2.32	0.11	2.17	2.47	B
C	83	2.45	0.11	2.30	2.59	BC
PL/I	40	2.57	0.15	2.35	2.78	BCD
SQL	42	2.58	0.15	2.37	2.78	BCD
$COBOL$	131	2.66	0.08	2.55	2.78	CD
$C++$	34	3.01	0.17	2.78	3.25	DE
$Java$	52	3.07	0.13	2.89	3.26	E

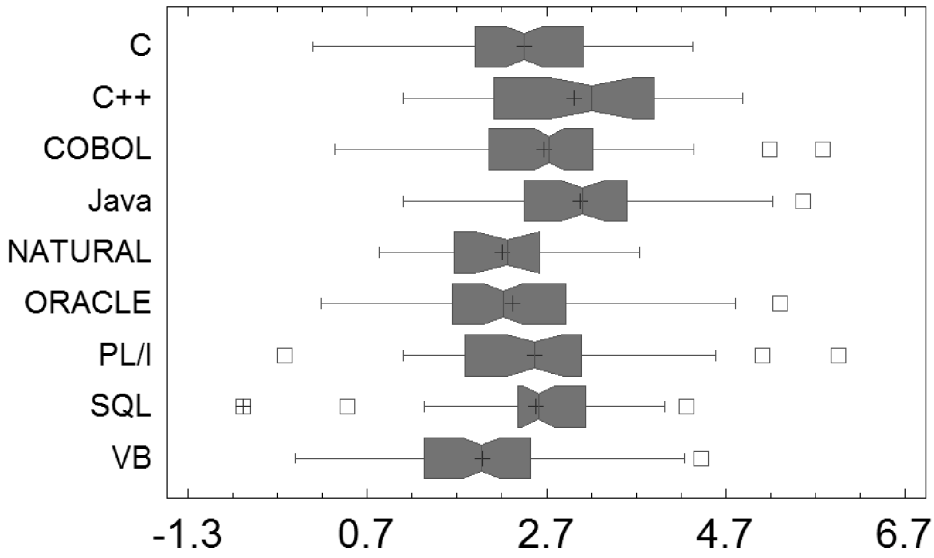

Figure 7: Box Plots of $ln(PDR)$ by Programming Language

COBOL and some of the projects reported as SQL). Less productive languages include C++ and Java. Perhaps surprisingly C++ is reported here as more productive than Java. Many of the Java projects recorded were carried out at the beginning of its adoption, perhaps showing a lack of experience when compared with C++. Another surprising result is that the mean productivity of C is a lot less than that of Java which in general is considered a difficult $3GL$ language. In general, languages considered as $4GL$, (database related languages, systems such as Oracle and SQL and many instances of VB) have better productivity values than those considered as third generation languages ($3GL$). It is worth noting that VB has not been consistently recorded as a third or fourth generation language in the ISBSG repository. A pairwise comparison of the differences between the primary programming languages is shown in Table 9.

The same procedure was followed for the *Development Platform* variable. As it can be observed from Figure 8 and Table 10, mainframe projects (MF) are less productive than the rest.

Finally, we also analysed the productivity of the different organisational types expecting some differences between them. However, as it can be observed in Figure 9 and Table 11, the different organisational types have similar productivity values with the exception of *Ordering* and *Voice Pro-*

Table 9: Contrasting $ln(PDR)$ Differences by Programming Language

Contrast	*Difference*	*±Limits*
C - C++	*-0.56	0.39
C - COBOL	-0.22	0.27
C - Java	*-0.62	0.34
C - NATURAL	0.24	0.52
C - ORACLE	0.13	0.30
C - PL/I	-0.12	0.37
C - SQL	-0.13	0.36
C - VB	*0.46	0.31
C++ - COBOL	0.35	0.37
C++ - Java	-0.06	0.42
C++ - NATURAL	*0.81	0.58
C++ - ORACLE	*0.69	0.39
C++ - PL/I	0.45	0.45
C++ - SQL	0.44	0.44
C++ - VB	*1.03	0.40
COBOL - Java	*-0.41	0.31
COBOL - NATURAL	0.46	0.51
COBOL - ORACLE	*0.35	0.27
COBOL - PL/I	0.10	0.35
COBOL - SQL	0.09	0.34
COBOL - VB	*0.69	0.28
Java - NATURAL	*0.87	0.55
Java - ORACLE	*0.76	0.34
Java - PL/I	*0.50	0.40
Java - SQL	*0.50	0.40
Java - VB	*1.09	0.35
NATURAL - ORACLE	-0.11	0.52
NATURAL - PL/I	-0.36	0.57
NATURAL - SQL	-0.37	0.56
NATURAL - VB	0.22	0.53
ORACLE - PL/I	-0.25	0.37
ORACLE - SQL	-0.26	0.36
ORACLE - VB	*0.33	0.31
PL/I - SQL	-0.01	0.42
PL/I - VB	*0.58	0.38
SQL - VB	*0.59	0.37

* denotes a statistically significant difference.

Table 10: Means for $ln(PRD)$ by *Development Platform*

	#	*Mean*	*Std Err*	*Lower limit*	*Upper limit*	*Homog. Groups*
$Multi$	23	1.84	0.19	1.57	2.11	A
PC	46	2.02	0.14	1.83	2.21	A
MR	59	2.25	0.12	2.08	2.42	A
MF	189	2.62	0.07	2.52	2.71	B

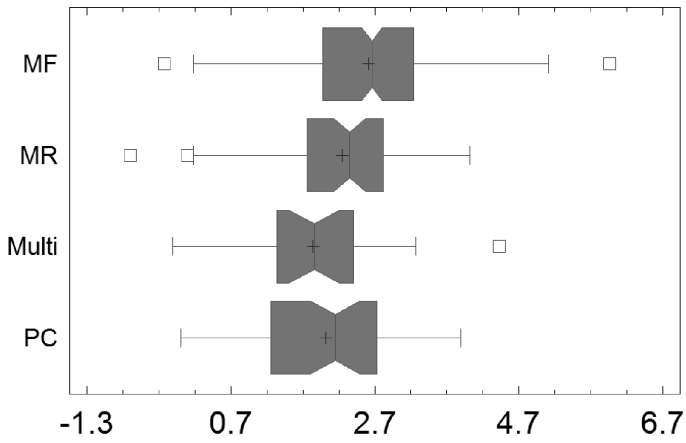

Figure 8: Box Plots of $ln(PDR)$ by *Development Platform*

visioning. In this case, *Voice Provisioning* seems to be a special case in the repository. All projects performed by a *Voice Provisioning* organisation were enhancement projects developed during 2002 and developed mainly in C++ (and some in C or Java) as primary programming language. Similarly, *Ordering* projects in our preprocessed dataset seem to be specific to a single organisation as all were developed during 2002, most in C or C++.

Therefore, contrary to what we were expecting, if we exclude the last two groups it seems that there are no statistical differences in productivity across different organisation types. The last two groups (ordering and voice provisioning) may have been submitted by a single organisation or they may reflect the real-time nature of their domains. In addition, those two organisation types do not have a large number of projects in the repository (approx. 9%).

In order to relate productivity to its most important related attributes, we used multiple regression analysis following Comstock et al. [11] approach. Using the most important variables when dealing with productivity reported by ISBSG, ATS, the primary programming language (instead of using the

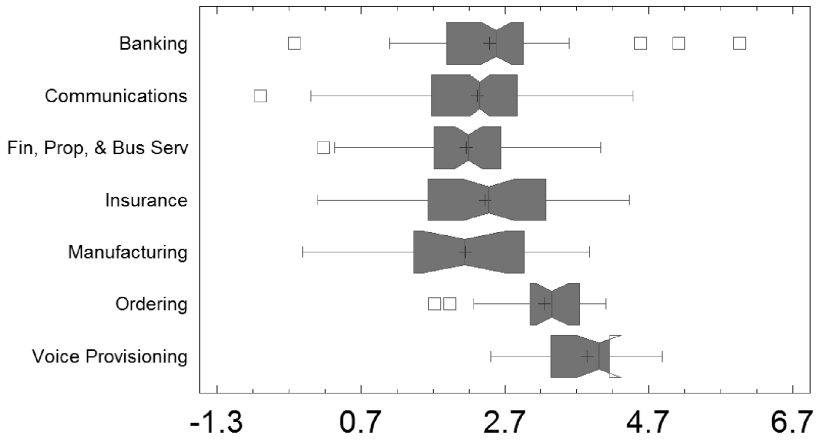

Figure 9: Box Plots of $ln(PDR)$ by *Organisation Type*

Table 11: Means for $ln(PRD)$ by *Organisation Type*

	#	Mean	Std Err	Lower limit	Upper limit	Homog. groups
Manufacturing	17	2.15	0.23	1.84	2.47	A
Fin, Prop & Business Serv	60	2.16	0.12	1.99	2.33	A
Communications	188	2.31	0.07	2.22	2.41	A
Insurance	51	2.42	0.13	2.24	2.60	A
Banking	65	2.48	0.12	2.32	2.64	A
Ordering	22	3.25	0.20	2.97	3.53	B
Voice Provisioning	17	3.85	0.23	3.53	4.16	C

Table 12: Coefficients of the Linear Regression for $ln(PDR)$

	Coefficient	*Std Error*	*t-value*	$Pr(>\|t\|)$
$log(ATS)$	0.57	0.12	4.69	$1.39e^{-05}$
$4GL$	-0.35	0.20	-1.73	0.08
MR	-0.42	0.27	-1.52	0.13
PC	-0.44	0.29	-1.48	0.14

primary programming language variable, we used *language type*) and *platform*, we obtained Eq. (1) considering the productivity as the independent variable and the other three variables as dependent.

$$log(PDR) = 1.85 + 0.57 \cdot log(ATS) + \alpha_i \phi(LangType_i) + \beta_j \phi(Platf_j) \quad (1)$$

where α_i and β_j correspond to the regression coefficients reported in Table 12.

The R^2 value is 32.33% (adjunsted R^2 is 0.28), which means that the model is acceptable with 32% of the variance in the dependent variable explained by the number of predictors. Table 12 also shows the the regression statistical results together with the $Pr(>|t|)$ value which shows that ATS is very relevant in predicting productivity (the smaller the value, the more relevant it becomes). The other values are also relatively small for software engineering standards. A problem with this analysis is that after removing instances with missing values we only have $4GL$ samples for the language type variable and PC and MR for the platform variable. Following Comstock et al. [11] the number of instances that remain to calculate the regression equation is 71, a bit below the borderline of the recommended $50 + 8k$ instances (where k is the number of predictors) [16]. We obtained similar results using the primary programming language and the maximum team size variables but in all cases team size is the variable explaining most of the R^2 value. From the point of view of project managers, a simple model as the one described here could be used by project managers for rough estimates when they do not have historical data as well as to understand the addition of personnel to productivity.

5. Threats to Validity

As with all empirical studies, there are some threats to validity that need to be considered in this study.

Construct validity is the degree to which the variables used in the study accurately measure the concepts they are supposed to measure. We have only used those projects classified with the quality attribute classified as A or B. Although there seems to be an agreement about the practical usefulness of publicly available repositories and a large amount of work using such datasets, the origin of the data is not completely known and therefore such a risk is present. There is also a risk in the way that the preprocessing of the dataset was performed (e.g., considering that all versions of Visual Basic are equivalent, unifying the consistency across the languages and their generation type, etc).

Internal validity is the degree to which conclusions can be drawn. A major threat can be related to the preprocessing performed. Although we have reported and specified each step, we may have discarded relevant attributes or joined values incorrectly. We also needed to deal with a large number of missing values for some of the attributes. For example, after the preprocessing, the maximum team size and average time size variables have almost 60% and 80% of missing values respectively with the risk of being very few or even a single organisation reporting such attributes. It could have been very valuable to know which projects correspond to a particular company, and other measures about personnel stability. The ISBSG does not report detailed information about the personnel characteristics and it is known that there are large differences in productivity. Therefore, projects of similar size can differ hugely in the amount of time and effort required. We are not able to analyse such characteristics with this repository. We believe that their usefulness is not clearly proven for some data mining and statistical analyses.

External validity is the degree to which the results of the research can be generalised to the population under study and other research settings. Although the data comes from real projects and is in principle, generalisable, the ISBSG organisation recognise these project may not be representative of the industry (they are selected by the companies providing the data which in turn belong to the better performed organisations with experience using function point estimation techniques). In the context of estimation, Kitchenham et al. [24] performed a Systematic Literature Review of predictions from cross-company models with predictions from within-company models. Although the authors state that their results are inconclusive as there is support in both directions, it seems to be easy to misuse cross-company models. In any case, each organisation should select and perform the studies and estimations with subsets of the data close to their domain and organisation

type, environment, language, etc.

6. Conclusions and Future Research Directions

In this paper, we analysed the ISBSG repository to study the relationship between staffing levels and productivity. We first preprocessed the data to obtain a sample of 951 instances for the ISBSG Release 10 from 4,105 projects. The preprocessing consisted of several steps such as selecting only projects reported with high quality according the ISBSG rules, removing attributes with a large percentage of missing values, homogenization of values (e.g., COBOL 2 and COBOL II) and removing typos and mistakes. For analyses requiring team sizes, only 199 out of the those 951 projects reported both values for maximum and average team sizes. All the remaining 951 projects were used for analyses that did not require the maximum and average team sizes (e.g., productivity of new vs. enhancement projects, the productivity of the programming languages or the productivity of organisation types).

The results showed that there are statistical correlations between team size, effort, productivity and project duration but these correlations are not always what one would expect from the literature or intuition. With the preprocessed data used in this study, projects with an average team size of 9 or more people (the threshold suggested in the literature) are less productive than those above such this threshold. Also enhancement projects have a better productivity than new projects. The type of language has a impact on productivity. Mainframe development is less productive than the other platforms. Voice provisioning organisations seem to be less productive than the others and are mainly developed in C++. The drawback of using this repository is the difficulty in cleaning and preprocessing of the data. There is a large number of attributes but many of them have a large number of missing values. These drawbacks can hinder its usefulness when applying statistical and data mining techniques and the reliability of its results.

In future work, further statistical and data mining studies will be performed with the ISBSG repository as well as other repositories in order to study the quality of such repositories, validate generic claims found in the literature and provide guidelines for decision making.

References

[1] T.K. Abdel-Hamid, The dynamics of software project staffing: A system dynamics based simulation approach, IEEE Transactions on Software

Engineering 15 (1989) 109–119.

[2] T.K. Abdel-Hamid, S.E. Madnick, Software Project Dynamics: An Integrated Approach, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[3] E. Alba, J. Chicano, Management of software projects with gas, in: Proceedings of the 6th Metaheuristics International Conference (MIC '05), Elsevier Science Inc., Vienna, Austria, 2005, pp. 13–18.

[4] E. Alba, J. Chicano, Software project management with gas, Information Sciences 177 (2007) 2380–2401.

[5] A.J. Albrecht, J.E. Gaffney, Software function, source lines of code, and development effort prediction: A software science validation, IEEE Transactions on Software Engineering 9 (1983) 639–648.

[6] G. Antoniol, A. Cimitile, G.A. Di Lucca, M. Di Penta, Assessing staffing needs for a software maintenance project through queuing simulation, IEEE Transactions Software Engineering 30 (2004) 43–58.

[7] G. Antoniol, M. Di Penta, M. Harman, Search-based techniques applied to optimization of project planning for a massive maintenance project, in: Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM '05), IEEE, Los Alamitos, California, USA, 2005, pp. 240–249.

[8] F.P. Brooks, The Mythical Man-Month, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, anniversary ed. edition, 1995.

[9] J. Chicano, F. Luna, A.J. Nebro, E. Alba, Using multi-objective metaheuristics to solve the software project scheduling problem, in: 13th Annual Conference on Genetic and evolutionary computation (GECCO'11), GECCO'11, ACM, New York, NY, USA, 2011, pp. 1915–1922.

[10] C. Comstock, Z. Jiang, J. Davies, Economies and diseconomies of scale in software development, Journal of Software Maintenance and Evolution: Research and Practice In Press (2011).

[11] C. Comstock, Z. Jiang, P. Naudé, Strategic software development: Productivity comparisons of general development programs, International Journal of Computer and Information Engineering 1 (2007) 486–491.

[12] M. Di Penta, M. Harman, G. Antoniol, The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study, Software: Practice and Experience 41 (2011) 495–519.

[13] M. Di Penta, M. Harman, G. Antoniol, F. Qureshi, The effect of communication overhead on software maintenance project staffing: a search-based approach, in: IEEE International Conference on Software Maintenance (ICSM 2007), pp. 315–324.

[14] J. Dolado, On the problem of the software cost function, Information and Software Technology 43 (2001) 61–72.

[15] J.B. Dreger, Function Point Analysis, Prentice Hall, NJ: Englewood Cliffs, 1989.

[16] S.B. Green, How many subjects does it take to do a regression analysis?, Multivariate Behavioral Research 26 (1991) 499–510.

[17] S. Gueorguiev, M. Harman, G. Antoniol, Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09), ACM, Montral, Canada, 2009, pp. 1673–1680 (Best Paper Award).

[18] M. Harman, B.F. Jones, Search-based software engineering, Information and Software Technology 43 (2001) 833–839.

[19] M. Heričko, A. Živkovic, I. Rozman, An approach to optimizing software development team size, Information Processing Letters 108 (2008) 101–106.

[20] IEEE, IEEE standard for software productivity metrics, 2003.

[21] IFPUG, Function Point Counting Practices Manual (CPM) - Release 4.3.1, International Function Point Users Group (IFPUG), 2010.

[22] ISBSG, Team Size Impacts Special Report, Technical Report, International Software Benchmarking Standards Group (ISBSG), 2007.

[23] B.A. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE-2007-01, Keele University, 2007.

[24] B.A. Kitchenham, E. Mendes, G.H. Travassos, Cross versus within-company cost estimation studies: A systematic review, IEEE Transactions on Software Engineering 33 (2007) 316–329.

[25] T. Kremmel, J. Kubalík, S. Biffl, Multiobjective evolutionary algorithm for software project portfolio optimization, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO'10), ACM, Portland, Oregon, USA, 2010, pp. 1389–1390.

[26] T. Kremmel, J. Kubalík, S. Biffl, Software project portfolio optimization with advanced multiobjective evolutionary algorithms, Applied Soft Computing 11 (2011) 1416–1426.

[27] P. Montgomery, Jr., A model of the software development process, Journal of Systems and Software 2 (1981) 237–255.

[28] NESMA, NESMA FPA Counting Practices Manual CPM 2.1, Netherlands Software Metrics Users Association (NESMA), 2004.

[29] P. Norden, Project life cycle modeling: Background and application of the life cycle curves, in: U.S. Army Computer Systems Command.

[30] P.V. Norden, Curve fitting for a model of applied research and development scheduling, IBM Journal of Research and Development 2 (1958) 232–248.

[31] P.C. Pendharkar, J. A.Rodger, An empirical study of the impact of team size on software development effort, Information Technology and Management 8 (2007) 253–262.

[32] M.D. Penta, M. Harman, G. Antoniol, F. Qureshi, The effect of communication overhead on software maintenance project staffing: a search-based approach, in: Proceedings of the 23rd IEEE International Conference on Software Maintenance (ICSM '07), IEEE, Paris, France, 2007, pp. 315–324.

[33] K. Pillai, V.S. Sukumaran Nair, A model for software development effort and cost estimation, IEEE Transtions on Software Engineering 23 (1997) 485–497.

[34] PMI, A Guide To The Project Management Body Of Knowledge (PMBOK Guides), Project Management Institute (PMI), 2004.

[35] L. Putnam, A general empirical solution to the macro software sizing and estimating problem, IEEE Transactions on Software Engineering 4 (1978) 345–361.

[36] P. Sentas, L. Angelis, I. Stamelos, G. Bleris, Software productivity and effort prediction with ordinal regression, Information and Software Technology 47 (2005) 17–29.

[37] R.K. Smith, J.E. Hale, A.S. Parish, An empirical study using task assignment patterns to improve the accuracy of software effort estimation, IEEE Transactions on Software Engineering 27 (2001) 264–271.

[38] A. Trendowicz, J. Münch, Factors influencing software development productivity – state-of-the-art and industrial experiences, volume 77 of *Advances in Computers*, Elsevier, 2009, pp. 185–241.

[39] K.G. van der Poel, S.R. Schach, A software metric for cost estimation and efficiency measurement in data processing system development, Journal of Systems and Software 3 (1983) 187–191.

[40] J. Verner, W. Evanco, N. Cerpa, State of the practice: An exploratory analysis of schedule estimation and software project success prediction, Information and Software Technology 49 (2007) 181–193.