World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:4, 2007

An Investigation on the Variation of Software Development Productivity

Zhizhong Jiang, Peter Naudé, and Craig Comstock

Abstract**—**The productivity of software development is one of the major concerns for project managers. Given the increasing complexity of the software being developed and the concomitant rise in the typical project size, the productivity has not consistently improved. By analyzing the latest release of ISBSG data repository with 4106 projects ever developed, we report on the factors found to significantly influence productivity, and present an original model for the estimation of productivity during project design. We further illustrate that software development productivity has experienced irregular variations between the years 1995 and 2005. Considering the factors significant to productivity, we found its variations are primarily caused by the variations of average team size for the development and the unbalanced use of the less productive development language 3GL.

Keywords—Development Platform, Function Point, Language, Productivity, Software Engineering, Team Size.

I. Introduction

SOFTWARE has become the key element in the evolution of computer-based systems and products. Over the past 50 years, software has evolved from a specialized problem solving and information analysis tool to an industry in itself [1]. The two primary problems in software development that have yet to be solved satisfactorily are making systems cost effective and of higher quality. A major obstacle to solve the problem of cost effective is the intrinsic complexity in developing software. Improving the productivity is an essential part of making system cost effective [2].

There have been two main directions on the study of productivity in software engineering literature. First, researches have been focused on the measure or estimation of productivity [3], [4], [5], [6], [7]. Second, emphasis has been laid on the discovery of methods or significant factors for productivity improvement [8], [9], [10], [11], [12], [13], [14].

With the increasing complexities and costs of software development, how to improve development productivity has been an ongoing concern for project managers. Unfortunately, despite the attention that has been given to it, the productivity of software development has not improved consistently. This needs to be seen in the light of the impressive improvements in hardware speed and network capacity [15]. Gross measures presented in the literature indicate that software productivity has been declining more rapidly than any other industry [16].

Whereas there is still no consensus that whether productivity has really declined [17], our analysis revealed that productivity has experienced irregular variations of decline and rise in the past decade, and there is no sign of imminent improvement. Based on the factors significant to productivity in our model, we found that its variations are primarily caused by the variations of average team size for development and the unbalanced use of the less productive development language 3GL.

Focusing on the analysis of the large database with 4106 projects developed worldwide, we organize this paper as follows. Section II gives an overview of the database and section III introduces the underlying factors significant to productivity; section IV and V are detailed procedures for model development; section VI presents full discussions on the derived model; section VII examines the goodness-of-fit for the model; sections VIII illustrates the variations in software development productivity between 1995 and 2005; sections IX and X display the variations of average team size and the unbalanced use 3GL over the years; section XI explains the factors leading to the productivity variation; and finally section XII presents the conclusion to the study.

This research was supported by the ISBSG (International Software Benchmarking Standards Group) for providing the data.

Zhizhong Jiang was with Department of Statistics, University of Oxford. He is now with University of Manchester, Booth Street West, Manchester, M15 6PB, UK (phone:+44(0)8708328157; fax: +44(0)1612756596; e-mail: Zhizhong.Jiang@postgrad.mbs.ac.uk, zhizhong.jiang@bnc.oxon.org).

Peter Naudé was with University of Bath, UK. He is now a Professor in Manchester Business School, University of Manchester, Booth Street West, Manchester, M15 6PB, UK (e-mail: pete.naude@mbs.ac.uk).

Craig Comstock was with Harvard University. He is now with University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD UK (e-mail: craig.comstock@lmh.ox.ac.uk).

II. Background

The common difficulty in the study of software metrics is the lack of accessible and reliable large dataset [18]. For the 18 major databases that were studied with productivity factors, Maxwell et al. [12] found 8 databases with sample size smaller than 50. Besides, many contemporary metrics repositories have limited use due to their obsolescence and ambiguity of documentation [19].

The data repository maintained by the International Software Benchmarking Standards Group (ISBSG) does not have the above deficiencies and has been widely researched [18], [20], [21], [22], [23]. The latest release of ISBSG data repository (Release 10) contains information on 4106 projects, and each project is recorded with up to 90 metrics or descriptive pieces of information. The manual accompanied with the data gives detailed descriptions of project attributes. The data repository is regularly updated with substantial projects added in every year. Our study will be focused on the analysis of data Release 10.

III. Data Description

Productivity is conceptualized as output produced per unit of input, and normally defined as project size divided by effort for the development. There have been diverse measures of productivity, for instance, function points divided by effort[5], [6], [24], [25], number of lines of code developed per unit of effort [12], [13], [26], case points divided by effort [7], number of models divided by effort [6], and number of tokens produced per person-month [27]. While the measure of function point has been criticized relating to its reliability [28] and usefulness of the complexity adjustments [29], it has been widely used to overcome the difficulties of traditional measure of lines-of- code in project planning and control [30].

In the ISBSG data repository there are several different counting techniques for function point (e.g., IFPUG, NESMA, Mark II). To have consistent measure, the functional size is adjusted by an adjustment factor and the resultant adjusted size is reported in Adjusted Function Points. For development effort it is recorded by two metrics: Summary Work Effort (total effort in hours spent on the project) and Normalized Work Effort[1]. Productivity is measured by the parameter Normalized Productivity Delivery Rate (PDR) which is calculated from Normalized Work Effort divided by Adjusted Function Points. Clearly PDR is an inverse measure of productivity in that the larger PDR, the smaller is the productivity.

Whereas data Release 10 contains many metrics recording each project developed, we only introduce those that likely have effects on productivity. Many of these metrics have been well studied before.

1. Average Team Size

It is the average number of people that worked on the project through the entire development process. ISBSG data also record another parameter Max Team Size, which is the maximum number of people that worked at any time on the project. We deem it more appropriate to use Average Team Size to assess the productivity level. Past studies suggest that productivity and team size are negatively associated [9],[12],[31],[32] [33].

2. Development Language

It defines the development language used for the project, including second generation languages (2GL), third generation languages (3GL), fourth generation languages (4GL) and Application Generator (ApG). In practice all 4GL languages are designed to reduce programming efforts, and they are more productive than 3GL languages [34]. Thus development language would be another latent factor significant to productivity.

[1] For projects covering less than a full development life-cycle, Normalized Work Effort is an estimate of the full development life-cycle effort. For projects covering the full development life-cycle, and projects where development life-cycle coverage is not known, this value is the same as Summary Work Effort.

3. Development Type

It describes whether the software development was a new development, enhancement or re-development. Development with enhancement may consume much of the total resources of programming groups and therefore does not necessary improve productivity [35].

4. Development Platform

It defines the primary development platform. Each project is classified as Mid-range, Mainframe, Multi-platform, or PC. Subramanian et al. [36] found platform has a significant effect on software development effort. This may indicate this factor is likely to affect development productivity.

5. Development Techniques

These are techniques used during software development (e.g. Waterfall, Prototyping). Some development techniques have been designed to expedite development. For instance, Rapid Application Development (RAD) was reported to significantly accelerate development [37], and prototyping was reported to yield products with about equivalent performance but with 45% less effort [38].

6. CASE Tool Used

It indicates whether the project used any CASE (Computer-Aided Software Engineering) tool. While Coupe and Onodu [39] regarded that CASE tool had a positive effect on productivity, a majority of organizations reported that CASE has not brought about a change in productivity[40]. Bruckhaus et al. [41] pointed out that the introduction of CASE tool does not necessarily improve productivity, and in certain situations it can actually decrease the productivity as it increases effort on specific activities.

7. Development Methodology

It describes how the development methodology was acquired. It can be Traditional, Purchased, Developed In-house, or a combination of Purchased and Developed. Liu and Mintram [18] found development methodology is not significant to effort, which is one of the determinants of productivity.

There are still three points that need to be mentioned here:

1) Since particular programming language (e.g. Java, C++) belongs to one of the generation languages (e.g. 3GL, 4GL), we did not consider the factor Primary Programming Language. Otherwise redundancy is introduced into the model to be developed.
2) Some scholar regarded project duration is significant to productivity, and productivity declines with increasing project duration [12]. However, we did not take this factor into account as our study is to explore the factors that intrinsically influence productivity. In fact, project duration is correlated with effort which is one of the two determining elements of productivity.
3) It is conceivable that senior software developers are more skillful and productive than junior developers. For instance, Kitchenham [42] observed there is significant

improvement in productivity when developers have high virtual machine experience and programming experience. However, ISBSG data repository does not record this. Hence we just assume the developers are all well-qualified practitioners.

After identification of the underlying factors significant to productivity, statistical analysis of the data will be conducted in the next three steps recommended by Maxwell [43]:

1) Validate the data and transform the variables;
2) Build the model;
3) Generalize findings from the model, and test the residuals and goodness-of-fit.

IV. Data Validation

The ISBSG data repository contains one parameter—Data Quality Rating, which indicates the reliability of the data recorded. It has four grades A, B, C, and D. While the data with quality ratings A, B and C are assessed as being acceptable, little credibility can be given to any data with rating D. Therefore, we excluded 141 projects with quality rating D.

Since our productivity is decided in part by functional size, the homogeneity of standardized methodologies for measuring functional size is essential. Among several different count approaches of function point, NESMA is considered to produce equivalent results with IFPUG [44]. In data Release 10, 3281 out of 4106 projects applied IFPUG as size count approach, and there are further 152 projects using NESMA. Thus, to give more reliable results, projects using size count approaches other than IFPUG and NESMA were excluded from the analysis.

Besides, projects with recording errors or unspecified information were removed. For instance, two projects were mistakenly recorded with Average Team Size 0.5 and 0.95 respectively. One project was recorded with development platform 'HH', and one project was recorded with unknown development methodology 'CICS'. Finally, the only one project that applied 5GL as development language was excluded.

After data cleaning there are 3322 projects remained. These data will be used for model development in the next section.

V. Model Development

For the metrics discussed in section III, PDR and Average Team Size are measured in ratio scale, and all the other six metrics are measured in nominal scale. We first examine the distributions of the two ratio variables PDR and Average Team Size. The two histograms in Fig. 1 (left half) indicate that the data are extremely skewed. We therefore take natural log transformation to redress the skewness for these two variables (right half of Fig. 1).

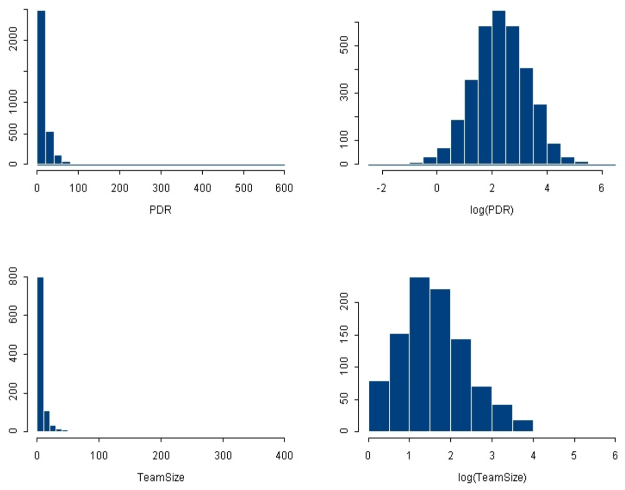

Fig. 1 Histograms of the two ratio variables PDR and Average Team Size, and their log-transformations

We then explore the potential relationship between PDR and Average Team Size after log transformation. Fig. 2 below is the simple scatterplot of log(PDR) against log of Average Team Size. Though they do not have a perfect positive linear relationship, the graph indicates that we can use linear model to approximate their relationship. Given that all other predictors are measured in ratio scale except Average Team Size, we can use multiple linear regression to fit a model with PDR as the dependent variable.

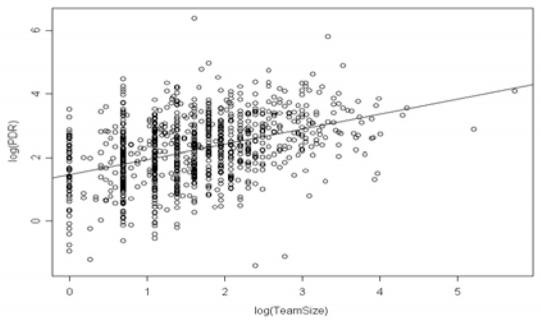

Fig. 2 The scatterplot of log(PDR) against log(TeamSize)

We first examine if there exists the problem of multicollinearity (strong correlations between predictor variables) in the data. That is, to see whether the use of some development method is likely to be associated with other techniques. The correlation tests indicated that there is no multicollinearity existent in the data. However, there are still two challenging difficulties to directly apply regression analysis to the data. First, for the metric Development Techniques there exist over 30 different techniques[2] in the data

[2] The ten primary techniques are Waterfall (643), Data Modelling (451), Process Modelling (294), JAD (Joint Application Development & Multifunctional Teams) (225), Prototyping (234), Regression Testing (164), Object Oriented Analysis & Design (143), Business Area Modelling (106), RAD (Rapid Application Development) (98), Event Modelling (85).

repository, and 766 projects even use various combinations of these techniques. Since the joint use of development techniques has the effect on productivity which is not simply additive of those effects when they are applied alone, we cannot ignore the effects of their interactions on the productivity. We managed to handle this difficulty by separating each of the ten main development techniques as one single binary variable with two levels indicating whether it is used or not (1 = used, 0 = not used), and taking into account all of the second-order interactions[3] between them. For the ten primary development techniques, Waterfall is the only one that is not in combination with any other techniques. As a result, 36 second-order interactions from the other nine development techniques are required for estimation. For all the uncommon development techniques, they are merged into one group labelled with 'Others'. The interactions between 'Others' and the main development techniques are not taken into consideration.

Second, substantial missing values are present in the 3322 observations after data cleaning. The metrics with large amount of missing data are Average Team Size (2349), Development Methodology (2068), Development Techniques (1891), CASE Tool Used (1899), Development Platform (853), and Development Language (447). The rule of thumb suggests a minimum sample size of 50+8*k* (*k* is the number of predictors) for multiple regression analysis [45]. Given that the factor Development Techniques itself has 47 (11 main effects and 36 interaction terms) predictors to estimate, a minimum sample size of 500 is required for regression. Therefore, if we add all the predictor variables simultaneously into the full model, then it only has a valid sample size of 302 which is not sufficient. Therefore, to treat this problem of enormous missing values, we added one indicator variable *Missing* which indicates whether particular cells of Development Techniques have missing data (1 = missing, 0 = not missing). In this way, 1891 projects with development techniques unrecorded are saved for regression testing. With this treatment the valid sample size became 402 which is acceptable but still low. The next subsection illustrates the steps we employed to overcome the deficiency of small valid sample size for multiple regression analysis. The names of the metrics are abbreviated, and the variables for regression are generalized in Table I below.

With the above two problems treated, the model is then developed in the following three steps.

Step 1): assess the significances of 48 variables concerning development techniques.

All the 48 predictors derived from the metric Development Techniques are kept in the regression model. These include 12 main effects (10 main development techniques, 2 additional variables *Others* and *Missing*) and 36 interaction terms. One of the 64 combinations of the six variables *TeamSize*, *Language*, *Platform*, *DevType*, *CASE*, and *Methodology* is then added into the model in turn, and the regression is performed. For unbalanced missingness, ANOVA (Analysis of Variance) based on Type I Sums of Squares depends on the orders that the terms are specified the model [46]. Therefore, for our study we used Type III Sums of Squares which does not depend on the orders of the terms specified in the model for unbalanced data. Repeating $2^6 = 64$ times, we found 19 interaction terms never showed significant based on their reported *p*-values (p-value < 0.05 is regarded as statistical significant [47], [48]). These 19 insignificant variables are excluded from the analysis, and the other 29 variables are retained for the next step. The removal of these 19 insignificant variables can reduce the sample size required for regression by 8×19=152.

TABLE I
DESCRIPTIONS OF VARIABLES FOR REGRESSION

Variable	Scale	Descriptions
PDR	Ratio	Normalized Productivity Delivery Rate.
TeamSize	Ratio	Average Team Size.
Language	Nominal	Development Language
DevType	Nominal	Development Type
Platform	Nominal	Development Platform
CASE	Nominal	CASE Tool Used
Methodology	Nominal	Development Methodology
Waterfall	Nominal	1= Waterfall, 0 = Not
Data	Nominal	1 = Data Modelling, 0 = Not
Process	Nominal	1 = Process Modelling, 0 = Not
JAD	Nominal	1 = JAD & Multifunctional Teams 0 = Not
Regression	Nominal	1 = Regression Testing, 0 = Not
Prototyping	Nominal	1 = Prototyping, 0 = Not
Business	Nominal	1 = Business Area Modelling, 0 = Not
RAD	Nominal	1 = Rapid Application Development 0 = Not
OO	Nominal	1 = Object Oriented Analysis & Design 0 = Not
Event	Nominal	1 = Event Modelling, 0 = Not
Others	Nominal	1 = uncommon development techniques 0 = Not
Missing	Nominal	1 = Missing, 0 = Not
	Nominal	36 interaction terms

Step 2): assess the significances of the six variables *TeamSize*, *Language*, *Platform*, *DevType*, *CASE*, and *Methodology*.

Keeping the 29 variables remained from step 1) in the model, we again add in one of the 64 combinations of *TeamSize*, *Language*, *Platform*, *DevType*, *CASE*, and *Methodology* by turns. We found variable *CASE* never exhibited significant among the 32 models that contain it. This indicates the use of CASE tool has no effect on productivity. Therefore, variable *CASE* is removed from the model. For the other five variables, *TeamSize* and *Language* frequently showed significant, whereas *Platform* moderately showed significant, and *DevType* and *Methodology* occasionally showed significant. These five variables are retained for further testing in step 3).

Step 3): stepwise remove insignificant terms with backward elimination and obtain final model.

[3] In regression analysis second-order interaction is the interaction between two variables, where *n*th-order interaction is the interaction among *n* variables. Since in data Release 10 a majority of the projects have missing values, it is intractable to assess high order of interactions which requires substantially large valid sample size. Therefore, we only considered second-order interaction.

After the above two steps we now have a valid sample size of 414 to test 44 predictors—4 levels of *Language*, 4 levels of *Platform*, 3 levels of *DevType*, 4 levels of *Methodology*, and 29 variables concerning Development Techniques retained from step 1). This sample size is just adequate for regression analysis based on the rule of minimum sample size 50+8*k* (*k* is the number of predictors). We add these 44 predictors all together into the model and perform multiple regression. We then exclude the predictor with the largest insignificant *p*-value reported, and fit the model with the predictors remained. Continuing this process we obtained the model with the final sets of significant terms. We will discuss the fitted model in detail in the next section.

VI. The Factors Significant to Productivity

The final model contains the predictors that are all significant (p-value < 0.05) to the dependent variable *PDR*. In other words, these predictors are factors essential to productivity. Table II displays the ANOVA table based on Type III Sums of Squares, and Table III presents the regression coefficients of the variables. The final model is fitted as follows:

$$\log(PDR) = 2.651 + 0.357\times\log(TeamSize) - 0.463\times I(3GL) - 1.049\times I(4GL) - 1.021\times I(ApG) - 0.138\times I(MR) - 0.219\times I(Multi) - 0.269\times I(PC) - 0.403\times I(OO) - 0.447\times I(Event) + 0.821\times I(OO{:}Event) - 0.276\times I(Business) - 0.024\times I(Regression) + 1.015\times I(Business{:}Regression)$$

TABLE II
ANOVA with Type III Sums of Squares for Regression

Regression Terms	Df	Sum of Sq	F Value	P-Value
log(TeamSize)	1	54.6	98.26	$< 10^{-15}$
Language	3	43.2	25.92	1.68×10^{-12}
Platform	3	5.2	3.13	0.0253
OO	1	0.0	0.00	0.9595
Event	1	0.0	0.06	0.8133
Business	1	1.1	2.04	0.1539
Regression	1	4.8	8.65	0.0034
OO : Event [4]	1	4.2	7.62	0.0060
Business: Regression	1	5.7	10.24	0.0015
Residuals	559	310.7		

[4] After data cleaning 29 projects used the combination of OO and Event Modeling, and 32 projects used the combination of Business Area Modeling and Regression Testing. The estimation for the interaction terms is reliable.

TABLE III
Regression Coefficients

Regression Terms	Coefficients
Intercept	2.651
log(TeamSize)	0.357
Language3GL	-0.463
Language4GL	-1.049
LanguageApG	-1.021
PlatformMR	-0.138
PlatformMulti	-0.219
PlatformPC	-0.269
OO	-0.403
Event	-0.447
Business	-0.276
Regression	-0.024
OO:Event	0.821
Business:Regression	1.015
Multiple R-Squared	0.36

To understand the model some interpretations are given as follows.

1) TeamSize denotes Average Team Size for the development. As one of the levels of the nominal variable *Language*, 3GL, 4GL, and ApG indicate which language is used for the development. The default language is 2GL. As one of the levels of the nominal variable *Platform*, MR (Mid Range), Multi (Multi-platform) and PC indicate which platform is used for the development. The default development platform is Mainframe. The remaining terms specify whether particular development technique is adopted for the development: OO (Object Oriented Analysis & Design), Event (Event Modeling), Regression (Regression Testing), and Business (Business Area Modeling).
2) log ( ) is the natural log transformation (with base e). The indicator function $I(\cdot)$ outputs only two values: value of 1 means the relevant technique in the parentheses is used, where value of 0 indicates not (That is, $I(a)=1$ if a is present, otherwise $I(\cdot)=0$). The operator **:** defines the second-order interaction between two variables. Hence there is no interaction if there is only one development technique used. That is, I(OO: Event) is 1 if and only if both OO and Event Modeling are used.
3) If the project adopts default language 2GL and default development platform Mainframe, then the model is reduced to:

$$\log(PDR) = 2.651 + 0.357\times\log(TeamSize) - 0.403\times I(OO) - 0.447\times I(Event) + 0.821\times I(OO{:}Event) - 0.276\times I(Business) - 0.024\times I(Regression) + 1.015\times I(Business{:}Regression)$$

In section III we mentioned PDR is defined as Normalized Work Effort divided by Adjusted Function Points. PDR is an inverse measure of the productivity in that the smaller PDR, the higher is the productivity. Examining the fitted model and the reported *p*-values for the variables in Table II, we can generalize:

1 Average Team Size, Development Language, Development Platform, and Development Techniques are significant factors for the productivity. Among these factors Average Team Size and Development Language are the two most essential ones with extremely significant *p*-values (<0.01%). In particular, we found Average Team Size explains 17.3% of the variance in log(PDR), and Development Language explains 7.8% of the variance in log(PDR). Therefore, Average Team Size is the most critical factor for productivity. On the other hand, the use of CASE Tool, Development Type, Development Methodology, and other unmentioned techniques have no considerable effects on the productivity.

2 Average Team Size and productivity are negatively associated. The increase of Average Team Size will lead to lower productivity.

According to the model, log(PDR) and log(TeamSize) have a positive linear relationship. Therefore, Average Team Size and productivity are negatively associated. This result is consistent with the findings by other researchers [12], [31], [32], [33]. Particularly, the double of Average Team Size will reduce productivity by 22% (1-exp(-0.357*ln2)).

3 Different development methods have varied significances to the productivity.

3.1 Languages 4GL and ApG are more productive than 3GL and 2GL;

3.2 Platform PC is slightly more productive than Multi-platform followed by Mid-range and Mainframe;

3.3 The single use of Event Modeling or Object Oriented Analysis & Design can lead to higher productivity. However, their joint use can only neutralize this effect;

3.4 The joint use of Business Area Modelling and Regression Testing is adverse to the improvement of productivity.

We notice the more negative of the coefficients of the indicator function I( ), the smaller the value of log(PDR), and hence the more productive of their corresponding development methods. With the default language 2GL and default platform Mainframe acted as benchmarks, the productivities of different development methods are compared by their matching coefficients of $I(\cdot)$.

For development languages the related coefficients of $I(\cdot)$ for 2GL, 3GL, 4GL and ApG are 0, -0.463, -1.049, and -1.021 respectively. So 4GL and ApG are more capable of reducing the value of log(PDR) than 2GL and 3GL. This means 4GL and ApG are more productive than 2GL and 3GL. This complies with the finding by Kitchenham [42] that significant productivity improvement is associated with the use of 4GL.

As for development platforms, platform PC (-0.269) is slightly more productive than Multi-platform (-0.219) which is moderately better than Mid-range (-0.138) with Mainframe acted as the default platform (0).

For development techniques, the single use of Event Modelling or Object Oriented Analysis & Design can reduce log(PDR) by -0.447 and -0.403 respectively. However, the result will be -0.029 (-0.447-0.403+0.821) if they are used together. This increases the value of log(PDR) and thus it loses the effect of improving productivity. On the other hand, the joint use of Business Area Modelling and Regression Testing will substantially enhance the value of log(PDR) to 0.715 (-0.276-0.024+1.015). Therefore, project managers should avoid using them together for the purpose of higher productivity.

VII. Model Validation

The primary objective of this study is to find the factors significant to productivity. This means the model we fitted is parsimonious with minimum number of predictors. While the saturated model contains all the predictors and has the most perfect goodness-of-fit, our parsimonious model was reported with multiple R^2 of 0.36. This indicates the fitted model is acceptable with 36% of the variance in the dependent variable explained by the minimum number of predictors.

Furthermore, in linear model it is assumed that the residuals are normally distributed with zero mean and homogeneity of variance [49]. Equal scatter of residual points about the horizontal axis indicates the residuals have homogeneity of variance [50]. We plotted the residuals against the fitted values, and found the points evenly scatter along the horizontal axis without obvious patterns. Therefore, the assumption of homogenous variance is validated.

Finally, we applied Kolmogorov-Smirnov test to test the assumption of normality of the residuals. The reported *p*-value is 0.5 which indicates the residuals do not deviate from normal distribution.

VIII. Variations in Productivity

Whereas there is still no consensus whether productivity has really declined [17], our findings revealed that productivity has actually experienced irregular variations of decline and rise between 1995 and 2005. Our analysis will be focused on the parameter Normalized Productivity Delivery Rate (PDR) which is an inverse measure of the productivity.

Since the yearly PDR data are highly skewed, natural log transformation is applied to rectify the skewness. The average PDR in each year is obtained by calculating the mean of log(PDR) and converting it back to PDR with exponential transformation. For some years the data deviate from normal even with log transformation. In these cases the median of the original PDR data is taken as the average PDR for that particular year. The stability of median justifies its use as a measure of centre when the data deviate from normal [51]. The annual averages of PDR between 1995 and 2005 are shown in

Table IV below. The column Number of Observations gives the number of projects recorded in the data repository for the related year. The sample sizes are all over 100 which are large to estimate the annual productivity. The trend of the average PDR is plotted in Fig. 3.

TABLE IV
THE AVERAGES OF PDR 1995-2005

Year	Average PDR	Number of Observations
1995	9.0	123
1996	10.4	102
1997	7.4	132
1998	7.7	264
1999	9.0	419
2000	9.3	544
2001	8.5	235
2002	10.5	335
2003	8.0	182
2004	8.2	257
2005	13.6	231

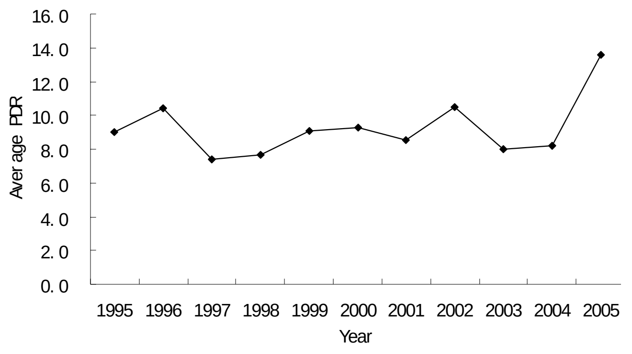

Fig. 3 The trend of annual average of PDR 1995-2005

The averages drawn in Fig. 3 demonstrate that the annual average PDR is unstable going through a series of changes. PDR rises in 1996, and then it suddenly drops in 1997 where the lowest PDR (7.4) occurs. Thereafter it continues to increase until year 2000. After that it varies considerably, dropping in 2001 but reaching its highest level to date in 2002. This is followed by two low years, before it reaches the peak (13.6) in 2005.

Recalling that PDR is an inverse measure of software development productivity, we can infer that productivity has experienced irregular variations of rise and decline over the past years. The year 1997 saw the highest productivity levels. However, and rather alarmingly, the lowest level of productivity arises in the latest year 2005 when, to deliver one function point, it actually needs 13.6 man hours. We therefore conclude that software development productivity has not been improving over time, and seek to explain this phenomenon.

IX. VARIATION OF AVERAGE TEAM SIZE

To explain the irregular variations in software development productivity, we can explore the factors which affect productivity. In section VI we identified that Average Team Size and Development Language are the two most significant factors influencing productivity. As a result, we turn to the study of these two factors.

Since the variable Average Team Size has a much skewed distribution, log transformation is taken. The average value is obtained either from the mean or median of the data depending on the validity of normal assumption. The annual average is displayed in Table V. Given the scarcity of the data in 2004 and 2005 (13 and 12 cases respectively), the data for these two years are merged with the new label 2004(5).

TABLE V
THE AVERAGE TEAM SIZE 1995-2005

Year	Average Team Size	Number of Observations
1995	3.0	63
1996	3.4	26
1997	3.0	61
1998	4.0	119
1999	5.0	219
2000	6.0	174
2001	6.3	61
2002	7.0	38
2003	4.6	38
2004(5)	6.2	25

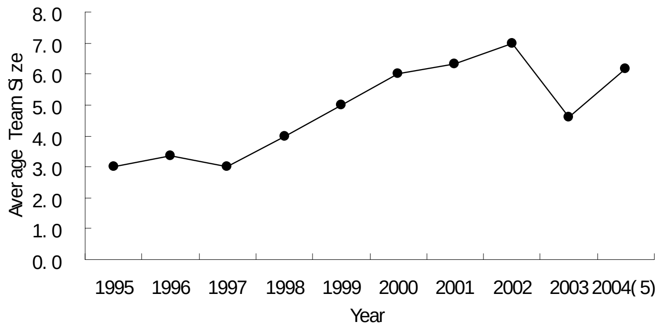

Fig. 4 The variation of Average Team Size 1995-2005

The data are further displayed in Fig. 4 which shows the extent to which Average Team Size has varied over time. It was very low in 1995 and 1997 (3.0). However, it constantly rises until the year 2002 where the maximum value to date emerges (7.0). After one remarkable drop in 2003, Average Team Size goes up again in 2004 and 2005. Clearly the overall trend is upwards. This confirms well with the view that due to the growing intricacy of software development, projects requires more and more developers to work together.

X. The Variable Uses of Development Languages

The second key factor essential to productivity is the development language. Most of the projects in the data Release 10 have adopted 3GL or 4GL as the development language. However, although 4GL has proved to be much more productive than 3GL [34], it has not been extensively used. Table VI below gives the breakdown of the four development languages used across the years.

TABLE VI
The Breakdown of the Uses of Four Language Types 1995-2005

Year	2GL	3GL	4GL	ApG	Total
1995	0	51	50	5	106
1996	0	48	18	8	74
1997	0	48	39	11	98
1998	1	117	68	10	196
1999	1	241	117	5	364
2000	3	329	187	14	533
2001	2	150	82	4	238
2002	0	262	84	1	347
2003	0	141	65	2	208
2004	0	139	162	0	301
2005	1	184	80	13	278

Table VI displays that the two development languages 2GL and ApG have rarely been used. As we have found in section VI, 4GL and ApG are both equally more productive than 3GL. Therefore, to see the effect of development language on the productivity variation, we can just examine the frequency of use of 3GL over the years. Fig. 5 below represents the percentage of 3GL used as the development language between 1995 and 2005. This indicates that there has been variable use of 3GL over the years. Year 2002 saw its most extensive use (75.5%). For most of the years over 60% the projects applied 3GL as the development language. Even in 1995, 1997 and 2004 there were still some 50% of the projects using 3GL (48.1%, 49.0% and 46.2% respectively). Therefore, we can conclude that 3GL has been the most prevalent development language in the past. Given its broad use in 2005 (66.2%) 3GL still remains popular today.

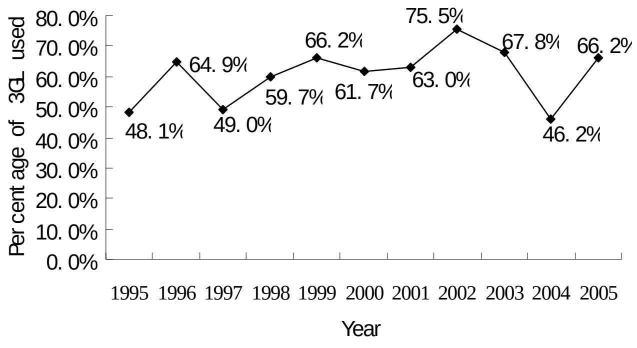

Fig. 5 The percentage of 3GL used 1995-2005

XI. The Explanation of the Productivity Variation

Given that Average Team Size and Development Language have been identified as the two key factors that influence productivity, the variations of productivity are most likely influenced by these two factors. We examine this by incorporating the preceding three figures into Fig. 6 below[5]. As we have mentioned Average Team Size has very sparse data for 2004 and 2005, and the resultant average is produced by fusing the data in these two years. To give systematic comparisons, we also acquired the average of PDR and 3GL respectively for these two years.

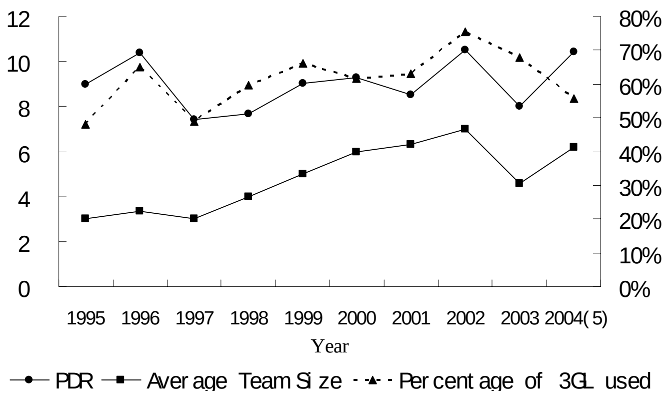

Fig. 6 The trend of PDR, Average Team Size and Percentage of 3GL used 1995-2005

Surprisingly the three curves in Fig. 6 demonstrate rather similar patterns. For most of the years PDR, Average Team Size and Use Percentage of 3GL have the same trends of rise or decline. This is consistent with the findings we obtained from the fitted model in section VI. The model shows that Average Team Size and PDR have a positive relationship, and 3GL is less productive than other development languages 4GL and ApG. Therefore, the growth of Average Team Size will lead to larger PDR (lower productivity), and the increasing use of 3GL can only result in lower productivity (larger PDR). In other words, the trend of productivity is decided by the congruous trends of Average Team Size and Use Percentage of 3GL.

[5] In Fig. 6 Average Team Size and PDR share the left y-axis and 3GL is represented by the right y-axis.

There are two periods of time (1999-2000, 2003-2004(5)) when the three curves have inconsistent trends. For both periods, PDR and Average Team Size increase while the Use Percentage of 3GL declines. As we have found from the model, Average Team Size is more critical than Development Language for productivity. Therefore, when Average Team Size and Use Percentage of 3GL move in opposite directions, productivity is decided in greater part by the former. However, other factors may also contribute to the productivity differences for these two periods.

Finally, we can identify one period (2000-2001) in which PDR declines while both Average Team Size and Use Percentage of 3GL slightly increase. This indicates that productivity has actually risen though it is expected to slightly go down. We contend this is caused by the accumulative effects of other moderately significant factors and insignificant factors for productivity. For instance, we can examine the moderately significant factor Development Platform for the two years 2000 and 2001. Table VII illustrates the usage of the four platforms for 2000 and 2001. In 2000 there are nearly two thirds (316/494) of the projects employed Mainframe and Mid-range, but in 2001 there were just 54% (134/246) of the projects using them. As indicated by the fitted model, Mid-range and Mainframe are least productive among the four platforms. Therefore, compared to 2001 the wide-scale uses of the less productive development platforms in 2000 restrained the improvement of the productivity.

TABLE VII
THE USES OF THE FOUR DEVELOPMENT PLATFORMS 2000-2001

Year	Mainframe	Mid-range	PC	Multi	Total
2000	215	101	134	44	494
2001	99	35	47	65	246

We now conclude that the variations of software development productivity in the past decade are largely caused by the variations of Average Team Size and the unbalanced use of the less productive language 3GL across the years.

XII. CONCLUSIONS

This study worked on the latest release of ISBSG data repository which is a very large database recording over 4000 software projects developed worldwide. Running multiple regression analysis this research found four factors significant to software development productivity. They are the two most essential factors Average Team Size and Development Language, and the other two moderately significant factors Development Platform and Development Techniques. A productivity evaluation model was presented for estimating productivity during project planning stage. The model revealed that the rise of Average Team Size for the development will decrease the productivity, and 3GL is less productive than other development languages 4GL and ApG.

In parallel with the mounting intricacy of software development, we found productivity has not improved over time but experienced irregular variations between 1995 and 2005, and there is no trace of its ongoing improvement. In view of the factors affecting productivity, we found Average Team Size and the uses of different development languages have also varied in the past. We identify that the variations in productivity are mainly caused by the variations of Average team size and the unbalanced use of the less productive language 3GL.

REFERENCES

[1] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. London: Mcgraw-Hill, 2001.
[2] S. T. Albin, The Art of Software Architecture: Design Methods and Techniques. New York: Wiley, 2003.
[3] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," *IEEE Transactions on Software Engineering*, vol. 30, pp. 1023-1035, 2004.
[4] W. S. Humphrey and N. D. Singpurwalla, "Predicting (individual) software productivity," *IEEE Transactions on Software Engineering*, vol. 17, pp. 196-207, 1991.
[5] K. D. Maxwell and P. Forselius, "Benchmarking software development productivity," *IEEE Software*, vol. 17, pp. 80-88, 2000.
[6] S. Morasca and G. Russo, "An empirical study of software productivity," presented at Proceedings of the 25th International Computer Software and Applications Conference on Invigorating Software Development, Chicago, 2001.
[7] M. Arnold and P. Pedross, "Software size measurement and productivity rating in a large-scalesoftware development department," presented at 20th International Conference on Software Engineering, Kyoto, Japan, 1998.
[8] N. R. Howes, "Managing software development projects for maximum productivity," *IEEE Transactions on Software Engineering*, vol. SE10, pp. 27-35, 1984.
[9] J. D. Blackburn, G. D. Scudder, and L. N. V. Wassenhove, "Improving speed and productivity of software development: a global survey of software developers," *IEEE Transactions on Software Engineering*, vol. 22, pp. 875-885, 1996.
[10] R. E. Loesh, "Improving productivity through standard design templates," *Data Processing*, vol. 27, pp. 57-59, 1985.
[11] G. R. Finnie, G. E. Wittig, and D. Petkov, "Prioritizing software development productivity factors using the analytic hierarchy process," *Journal of Systems and Software*, vol. 22, pp. 129-139, 1993.
[12] K. Maxwell, L. V. Wassenhove, and S. Dutta, "Software development productivity of European space, military and industrial applications," *IEEE Transactions on Software Engineering*, vol. 22, pp. 706-718, 1996.
[13] D. N. Card, F. E. McGarry, and G. T. Page, "Evaluating software engineering technologies," *IEEE Transactions on Software Engineering*, vol. SE-13, pp. 845-851, 1987.
[14] B. W. Boehm and P. N. Papaccio, "Understanding and Controlling Software Costs," *IEEE Transactions on Software Engineering*, vol. 14, pp. 1462-1477, 1988.
[15] I. R. Chiang and V. S. Mookerjee, "Improving software team productivity," *Communications of the ACM*, vol. 47, pp. 89-93, 2004.
[16] D. Anselmo and H. Ledgard, "Measuring productivity in the software industry," *Communications of the ACM*, vol. 46, pp. 121-125, 2003.
[17] R. Groth, "Is the software industry's productivity declining?," *IEEE Software*, vol. 21, pp. 92-94, 2004.
[18] Q. Liu and R. C. Mintram, "Preliminary data analysis methods in software estimation," *Software Quality Journal*, vol. 13, pp. 91-115, 2005.
[19] W. Harrison, "A flexible method for maintaining software metrics data: a universal metrics repository," *Journal of Systems and Software*, vol. 72, pp. 225-234, 2004.
[20] C. J. Lokan, "An empirical analysis of function point adjustment factors," *Information and Software Technology*, vol. 42, pp. 649-660, 2000.
[21] R. Jeffery, M. Ruhe, and I. Wieczorek, "A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data," *Information and Software Technology*, vol. 42, pp. 1009-1016, 2000.

[22] J. J. Cuadrado-Gallego, M. Sicilia, M. Garre, and D. Rodríguez, "An empirical study of process-related attributes in segmented software cost-estimation relationships," *Journal of Systems and Software*, vol. 79, pp. 353-361, 2006.
[23] J. Moses, M. Farrow, N. Parrington, and P. Smith, "A productivity benchmarking case study using Bayesian credible intervals," *Software Quality Journal*, vol. 14, pp. 37-52, 2006.
[24] G. H. Subramanian and G. E. Zarnich, "An examination of some software development effort and productivity determinants in ICASE tool projects," *Journal of Management Information Systems*, vol. 12, pp. 143-160, 1996.
[25] C. A. Behrens, "Measuring the productivity of computer systems development activities with function points," *IEEE Transactions on Software Engineering*, vol. SE-9, pp. 648-652, 1983.
[26] A. MacCormack, C. F. Kemerer, M. Cusumano, and B. Crandall, "Trade-offs between productivity and quality in selecting software development practices," *IEEE Software*, vol. 20, pp. 78-85, 2003.
[27] M. H. Halstead, *Elements of Software Science*. New York: Elsevier, 1977.
[28] C. F. Kemerer, "Reliability of function points measurement: a field experiment," *Communications of the ACM*, vol. 36, pp. 85-97, 1993.
[29] C. R. Symons, "Function point analysis: difficulties and improvements," *IEEE Transactions on Software Engineering*, vol. 14, pp. 2-11, 1988.
[30] C. F. Kemerer and B. S. Porter, "Improving the reliability of function point measurement: an empirical study," *IEEE Transactions on Software Engineering*, vol. 18, pp. 1011-1024, 1992.
[31] F. Louis, "Team size and productivity in systems development," *Information Systems Management*, vol. 8, pp. 27-35, 1991.
[32] S. D. Conte, H. E. Dunsmore, and Y. E. Shen, *Software Engineering Metrics and Models*. Redwood City, CA: Benjamin-Cummings Publishing, 1986.
[33] E. Mendes and B. Kitchenham, "Web Productivity Measurement and Benchmarking," in *Web Engineering*, E. Mendes and N. Mosley, Eds. Berlin: Springer, 2006, pp. 75-106.
[34] R. Klepper and D. Bock, "Third and fourth generation language productivity differences," *Communications of the ACM*, vol. 38, pp. 69-79, 1995.
[35] B. P. Lientz, E. B. Swanson, and G. E. Tompkins, "Characteristics of application software maintenance," *Communications of the ACM*, vol. 21, pp. 466-471, 1978.
[36] G. H. Subramanian, P. C. Pendharkar, and M. Wallace, "An empirical study of the effect of complexity, platform, and program type on software development effort of business applications," *Empirical Software Engineering*, vol. 11, pp. 541-553, 2006.
[37] J. Martin, *Rapid Application Development*. New York: Macmillan, 1991.
[38] B. W. Boehm, T. E. Gray, and T. Seewaldt, "Prototyping vs. specifying: A multi-project experiment," presented at 7th International Conference on Software Engineering, Orlando, 1984.
[39] R. T. Coupe and N. M. Onodu, "An empirical evaluation of the impact of CASE on developer productivity and software quality," *Journal of Information Technology*, vol. 11, pp. 173-181, 1996.
[40] D. Flynn, J. Vagner, and O. D. Vecchio, "Is CASE technology improving quality and productivity in software development?," *Logistics Information Management*, vol. 8, pp. 8-23, 1995.
[41] T. Bruckhaus, N. H. Madhavii, I. Janssen, and J. Henshaw, "The impact of tools on software productivity," *IEEE Software*, vol. 13, pp. 29-38, 1996.
[42] B. A. Kitchenham, "Empirical studies of assumptions that underlie software cost-estimation models," *Information and Software Technology*, vol. 34, pp. 211-218, 1992.
[43] K. Maxwell, *Applied Statistics for Software Managers*. New Jersey: Prentice Hall, 2002.
[44] NESMA, NESMA FPA Counting Practices Manual 2.0: Nesma Association, 1996.
[45] S. A. Green, "How many subjects does it take to do a multiple regression analysis?," *Multivariate Behavioral Research*, vol. 26, pp. 499-510, 1991.
[46] SAS Institute Inc, *SAS/Stat User's Guide: Version 6* 4th edition ed. Cary, NC: SAS Institute Inc, 1990.
[47] M. J. Crawley, *An Introduction to Data Analysis using S-Plus*. Chichester: John Wiley & Sons, 2002.
[48] W. N. Venables and B. D. Ripley, *Modern applied statistics with S*. New York: Springer, 2002.
[49] A. C. Rencher, *Linear Models in Statistics*. New York: John Wiley & Sons, 2000.
[50] W. J. Krzanowski, *An Introduction to Statistical Modelling*. London: Arnold, 1998.
[51] R. Peck, C. Olsen, and J. Devore, *Introduction to Statistics and Data Analysis*. London: Duxbury, 2001.