An Empirical Study To Revisit Productivity Across Different Programming Languages

Yingling Li[✤†], Lin Shi[✤†], Jun Hu[✤†], Qing Wang[✤┤†], Jian Zhai[✤†]

[✤]Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing, China

[┤] State Key Laboratory of Computer Sciences, Institute of Software, Chinese Academy of Sciences, Beijing, China

[†]University of Chinese Academy of Sciences , Beijing, China

{yingling, shilin, hujun, wq, zhaijian}@itechs.iscas.ac.cn

Abstract—**The development of High-level programming languages(HLPL) is a long process of evolution, which has gone through procedure-oriented languages, object-oriented languages, script languages and visual & database languages. Throughout the process of evolution, coding in an efficient and convenient way is the primary impetus. Hence, the productivity should vary across different languages. When evaluating the productivity of developers coding in different programming languages, such variations are usually ignored, which may not reflect their actual coding efficiencies and make the developers feel unfair. Especially, ignoring the variations will lead to inappropriate baselines of process performance, thus may potentially reduce the effectiveness of quantitative management. In this paper, we conducted an empirical study to revisit the productivity variations across different programming languages based on the data of International Software Benchmarking Standards Group (ISBSG) and a software organization. We found that, in most language categories, the productivity is all significantly different in ISBSG and the organization respectively. In the software industry represented by ISBSG, the relative productivity levels are gradually increasing with the evolution of HLPL. However, for the organization, it does not always keep the same increasing trend, but the average productivity within the four categories is almost stable. The finding could guide the software organization to establish an appropriate productivity baseline.**

Keywords—Productivity variations; HLPL; PO; OO; SL; VD; Function Points; Lines of code.

I. INTRODUCTION

Productivity is the most fundamental and crucial measurement to access the production efficiency of software organizations. It can be used for tactical reasons such as cost estimation and project scheduling. It can also be used to evaluate the capability of developers to produce software. Existing studies show that the average size of function points is significantly different across languages [3]. When comparing the productivity of developers who code in different languages, the impact of programming languages should be taken into consideration. Otherwise, the collected productivity may not reflect their actual performances. For example, two developers spent the same time on two tasks. One developer produced 500 lines of code for Linux Kernel optimization in C language, while another developer produced 5000 lines of code for web UI in HTML language. If the productivity variations between C and HTML language are not considered, the productivity of the second developer is 10 times higher than the first one. But, indeed, the programming performances and the contributions of these two developers might be quite similar.

In fact, the programming languages have evolved into more convenient and effective versions since the first generation languages. More than 1000 programming languages have been invented. Do some of them have approximate productivity? If we classify the languages with approximate productivity into the same group, will the productivity be significantly different across groups? Addressing the above concerns is the primary purpose of this study. In this paper, we performed an empirical study on the ISBSG dataset that contains productivity data from a large number of software companies, and a real world process dataset, which contains the productivity data of an individual organization−−the research and develop center in Institute of Software Chinese Academy of Sciences (RDCIOS). We divided HLPL into four categories based on their evolution, and explored their productivity differences. We performed the empirical study based on three research questions as follows:

RQ1: Whether the productivity for the four language categories is significantly different?

- RQ1.1: Is the productivity across the four language categories significantly different in the software industry and the individual organization respectively?
- RQ1.2: Is the productivity within each language category significantly different in the software industry and the individual organization respectively?

RQ2: What are the relative productivity levels of the four language categories for the software industry and individual organizations respectively?

RQ3: Regarding an individual organization, can the productivity performance baseline be established based on the four categories to support statistical process control?

The main contributions of this paper are as follows:

(I) Our study shows that the programming productivity varies with the evolution of HLPL in ISBSG and RDCIOS. Within each language category, the productivity of most programming languages has no significant difference. The results can indicate which languages are comparable, and which are not. Such differences also show that the four categories of HLPL are reasonable.

(II) In the scope of software industry, the relative productivity levels are gradually increasing with the evolution of HLPL. However, for the organizations, they don't always keep the same increasing trend. It means that the organizations

978-1-5386-3681-7/17 $31.00 © 2017 IEEE
DOI 10.1109/APSEC.2017.60

should establish the productivity baselines based on their own data, rather than simply adopt the industry baseline.

(III) We present an example about how to build the productivity baselines based on the four new categories. The generated baselines are useful to improve the quantitative management for organizations.

The rest of this paper is organized as follows. Section II presents the experiment setup. Section III shows the empirical analysis, Section IV presents the possible threats. Section V introduces related work. Section VI concludes and discusses future work.

II. Experiment setup

In this section, we introduce the subjects of the study, the categories of programming languages, the metrics of productivity, and the process of data preparation.

A. Subjects

The subjects of the study include the software industry ISBSG and the individual organization RDCIOS.

ISBSG dataset is one of the most extensive SE datasets. It contains productivity data from software development projects of different countries and different domains. In China, there is a non-profit organization, called Beijing Software Cost Evaluation Alliance(BSCEA), which is responsible for collecting data from Chinese software industry. We referred the combined data from ISBSG and BSCEA as ISBSG. We used ISBSG as the industry data.

Besides industry dataset from ISBSG, we also investigated one individual organization − − RDCIOS. RDCIOS uses SmartDev, which is a software development support platform developed by RDCIOS, to manage and monitor their projects. There are more than 300 developers collaborating via this platform and more than 100 projects are maintained in SmartDev.

B. Programming Language Categories

Programming languages have evolved from the first generation into the fifth generation [8]. The first generation languages (1GL) are called machine languages. The second generation languages (2GL) are assembly languages. The third languages (3GL) belong to high-level languages, which include hundreds of different languages that are widely used by programmers nowadays. The fourth generation languages (4GL) refer to the languages that aim to provide a higher level of abstraction to the internal computer hardware details. Moreover, some researchers proposed the fifth generation languages(5GL), which are mainly used in artificial intelligence research to build specific programs. But they are still in the research stage.

Obviously, the productivity of 3GL and 4GL is much higher than 1GL and 2GL [8] [10]. In practices, the boundary between the 3GL and 4GL is quite blurry. The 4GL inherited most of characteristics of 3GL. While some advanced 3GL like Python, Ruby, and Perl, combined some features of 4GL in the general-purpose environment of 3GL. Libraries with the features of 4GL have been developed for most popular 3GL.

In our study, we focused on investigating the productivity differences both in 3GL and 4GL languages. HLPL mainly refer to the 3GL and 4GL[7,9]. Considering the blurred boundary between 3GL and 4GL, we revisited the programming languages by classifying them into four new categories based on the evolution of HLPL [7, 8, 10]:

- Procedure-Oriented (PO) Languages: the early 3GL languages are used for writing programs that are more or less independent from a particular type of computer, called procedure-oriented languages. This category includes Basic, C, COBOL, FORTRAN, etc. The PO languages belong to compiled languages[9];
- Object-Oriented (OO) Languages: after PO languages, OO languages emerged as the 3GL, which are used for supporting a programming paradigm based on the concept of “object”, such as: C++, Java, C# and Ada. The OO languages belong to compiled languages too;
- Script Languages (SL) Languages: script languages are designed to support scripting special run-time environments that can automate the executions of tasks. They already have some features of 4GL, for example, they don’t need to be compiled to produce results. Javascript, Python, Ruby, Perl are popular used. The SL languages usually refer to interpreted languages[9,10].
- Visual and Database (VD) Languages: The VD languages are designed for facilitating the development of web page and the access to database, such as T-SQL, HTML, JSP, etc.

C. Metrics

In ISBSG, the size of software projects is measured by Function Point (FP), and the development effort is measured by Personal Hours (PH). Therefore, we define the productivity P_{fp} as follows.

$$P_{fp} = \frac{\sum \text{developing effort}}{\sum \text{funtion points}} \tag{1}$$

In RDCIOS, the size of software projects is measured by lines of code, and the related time is recorded. Therefore, we define the productivity P_{loc} as follows, t denotes the given time, such as an hour, a week.

$$P_{loc} = \frac{\sum \text{code lines}}{\text{t}} \tag{2}$$

Although the equations of productivity are different for the two datasets, both metrics can reflect the level of productivity for each dataset. It is safe to draw conclusions towards productivity differences based on these two metrics.

D. Data Preparation

In this section, we describe how we collect and prepare data from ISBSG and RDCIOS.

1) ISBSG

The dataset of ISBSG covers a wide variety of the software industry, which spans 27 years, including 8,113 projects. 97.5% projects could measure the productivity by P_{fp}, and most projects were developed by simple language. The information of the original dataset is shown in TABLE I.

The process of data collection includes two steps. First, we excluded the data according to the two criteria: (I) projects that are rated as C or lower score in terms of data quality levels that are provided by ISBSG [5]. (II) projects that have abnormal productivity ($P_{fp} > 100$).

TABLE I. THE DATASETS INFORMATION OF ISBSG

Original Dataset		Filtered Dataset	
Time Span	1989-2016	PO	1218
Dataset Size	8113	OO	1486
Projects	8113	SL	191
Languges	> 100	VD	606
		Sum	3501

TABLE II. THE DATASETS INFORMATION OF RDCIOS

Original Dataset		Filtered Dataset	
Time Span	2016.8-2017.5	PO	100
Dataset Size	4087commits	OO	210
Projects	181	SL	379
Languges	12	VD	51
		Sum	740

According to the exclusion criteria, 56.8% projects are filtered out. We further divided the data into four groups according to the four language categories introduced in section II-B. The final dataset contains 3501 records as shown in TABLE I. Each record denotes the productivity of each project, measured by hours per function point (PH/FP).

2) RDCIOS

In RDCIOS, the developers submitted their codes to SmartDev via commits. Developers recorded the time they spent on each commit by appending the effort expression (e.g., "effort=8") to the commit message. SmartDev can automatically collect effort data from commit messages. By leveraging this functionality, we can conveniently collect the productivity data of each commit.

The process of data collection is as follows: First, we selected the projects, and collected the commit data to obtain the original dataset. We selected the projects based on two criteria: (I) highly active projects; (II) the source codes are well-managed in SmartDev.

Then we collected all the commits of the selected projects from Git repository by executing 'git diff', and collected the effort data for each commit. We only collected the commits that contain the effort information. Because the productivity is measured based on codes in RDCIOS, we also excluded the commits that mainly contain documents, and quite few codes. In total, 4087 commits were collected as shown in TABLE II.

After obtaining the original dataset, we divided the dataset into four language categories, and calculated the individual productivity according to the Equation (2). The final dataset contains 740 records. Each record denotes the lines of code of individuals per hour (Loc/PH).

III. EMPIRICAL AYALYSIS

In this section, we present the empirical analysis on the three research questions.

A. RQ1: Whether the productivity for the four language categories is significantly different?

We answer the RQ1 from two aspects: whether the productivity differences across the four categories and within each category are statistically significant.

Since the distribution of datasets is not normal, we apply the nonparametric Kruskal-Wallis test and Mann-Whitney test to check in ISBGSG and RDCIOS respectively. We use 95% as the confidence level.

TABLE III. THE RESULTS OF THE PAIRED-SAMPLES TEST IN ISBSG

Group	PO	OO	SL	VD
PO	NA	0.0000	0.0000	0.0000
OO			0.0110	0.0000
SL				0.0346
VD				NA

TABLE IV. THE RESULTS OF THE PAIRED-SAMPLES TEST IN RDCIOS

Group	PO	OO	SL	VD
PO	NA	1.0000	**0.0000**	0.2135
OO			**0.0000**	0.2135
SL				0.2031
VD				NA

[a] The statistically significant results are highlighted.

1) RQ1.1: Is the productivity across the four language categories significantly different in the software industry and the individual organization respectively?

a) ISBSG

We first performed the Kruskal-Wallis test to check whether the productivity differs with different categories in ISBSG. The p-value equals to 0.00, which means that the productivity differences across the four categories are statistically significant.

Then we used the Mann-Whitney test to investigate whether productivity between any paired categories is significantly different. The results are shown in TABLE III. We can see that the p-value for each two paired categories is less than 0.05. Therefore, the productivity is significantly different between any two categories.

b) RDCIOS

Similarly, we first checked whether the productivity differs with different categories. The p-value equals to 0.00, which means the productivity differences are significant across the four categories in RDCIOS. Then we performed the paired-samples test, and the results are shown in TABLE IV. We observed that only the SL category significantly differs with the PO and OO categories. The VD category shows no difference with any other categories.

Finding 1.1: regarding the four language categories, the programming productivity is significantly different across the four categories in ISBSG, but the SL category shows significant differences with the PO and OO categories in RDCIOS. It shows that the individual differences exist. If they simply use industry benchmark to conduct effort estimation without self-adjusted, the estimation variation will be too large to control.

2) RQ1.2: Is the productivity within each language category significantly different in the software industry and the individual organization respectively?

a) ISBSG

We first carried out the Kruskal-Wallis test. The results are shown in TABLE V.

TABLE V. THE DIFFERENCES OF PRODUCTIVITY WITHIN EACH LANGUAGE CATEGORY IN ISBSG

Item	PO	OO	SL	VD
H	25.57	132.05	26.92	65.21
p-value	**0.000**	**0.000**	**0.008**	**0.000**

[a] The statistically significant results are highlighted.

TABLE VI. THE RESULTS OF THE PAIRED-SAMPLES TEST IN ISBSG

PO						
Language	***Cobol***	***Easytr-ieve***	***Fortran***	***Pascal***	***PL/I***	***ABAP***
C	0.9345	0.1111	0.2571	**0.0025**	**0.0025**	**0.0140**
Cobol		0.1022	0.2711	**0.0216**	**0.0000**	**0.0052**
Easytrieve			1.0000	0.3429	0.5363	**0.0091**
Fortran				0.0828	0.8677	**0.0386**
Pascal					0.0472	**0.0073**
PL/I						**0.0000**
ABAP						NA

SL							
Language	***Java-Script***	***Natural***	***Perl***	***PHP***	***Python***	***Shell***	***Unix Shell***
Groovy	0.2150	0.3260	0.052	0.1290	0.8170	0.973	**0.031**
JavaScript		0.0050	0.020	0.0050	0.1990	**0.015**	**0.003**
Natural			0.140	0.1350	0.643	0.086	0.224
Perl				0.6130	1.0000	0.050	0.207
PHP					1.0000	**0.035**	0.57
Python						0.720	1.00
Shell							**0.012**
Unix shell							NA

OO						
Lang-uage	***C++***	***Java***	***Power Builder***	***Visual Basic***	***Visual Studio .Net***	***Visual FoxPro***
C#	**0.0270**	**0.000**	**0.0006**	**0.0010**	**0.010**	**0.0100**
C++		**0.007**	0.2322	0.0650	0.500	0.0700
Java			0.8643	0.3490	0.260	0.0900
Power Builder				0.7150	0.150	0.0500
Visual Basic					0.390	0.1200
Visual Studio .Net						0.150
Visual FoxPro						NA

VD								
Lang-uage	***ASP .Net***	***Html***	***Ingres***	***JSP***	***Oracle***	***PL/ SQL***	***Pro*C***	***SQL***
ASP	0.460	0.460	0.170	0.076	0.117	0.3109	**0.0052**	**0.008**
ASP .Net		0.830	1.000	0.234	0.721	0.1775	0.5387	0.651
Html			0.490	0.062	0.938	0.0899	0.1741	0.455
Ingres				0.079	0.458	0.1554	0.3579	0.920
JSP					**0.030**	0.2917	0.0413	0.037
Oracle						**0.0001**	**0.0340**	**0.015**
PL/SQL							**0.0073**	**0.000**
Pro*C								0.369
SQL								NA

[a] The statistically significant results are highlighted

As shown in TABLE V. , the p-value is less than 0.05 in the four categories, which means that productivity differences within each language category are significant. Therefore, at least one language within each category differs with other languages. We further carried out the paired-samples test by using the Mann-Whitney test to find out the root cause. The detail results are shown in TABLE VI.

In the PO category, we observed that the productivity of ABAP differs with other languages. This can be explained that the variation tested by using the Kruskal-Wallis test is mainly caused by ABAP. ABAP is not a popular language, which is designed primarily for business purposes. Most of other languages show no significant difference with each other in the PO category. In the OO category, we found that the productivity variation is mainly caused by C#. In the SL and VD categories, the p-value for most paired languages within each category is greater than 0.05, which means that the productivity is not significantly different between most paired languages in these two categories.

Based the above statistical test, we can see that the four categories can distinguish HLPL by productivity.

b) RDCIOS

We first carried out the Kruskal-Wallis test. The results are shown in TABLE VII. The dataset of the PO category contains only C language, and cannot perform the Kruskal-Wallis test and Mann-Whitney test.

We observed that the p-value is less than 0.05 in the other three categories, which means that there are large productivity differences of different languages within each category. At least one language differs with the others. Therefore, we further carried out the paired-samples test to find out the root cause.

TABLE VII. THE DIFFERENCES OF PRODUCTIVITY WITHIN EACH LANGUAGE CATEGORY IN RDCIOS

Item	**OO**	**SL**	**VD**
H	6.52	49.91	9.13
p-value	**0.011**	**0.000**	**0.010**

[a] The statistically significant results are highlighted

TABLE VIII. THE RESULTS OF THE PAIRED-SAMPLES TEST OF THE SL CATEGORY IN RDCIOS

Language	**RDCIOS_SL**					
	IML	***Javascript***	***PHP***	***Python***	***Ruby***	***Shell***
IML	NA	0.1438	0.1887	0.2409	0.4620	0.9614
Javascript			0.5472	0.0431	**0.0272**	**0.0000**
PHP				0.1415	0.0417	**0.0000**
Python					0.2261	**0.0000**
Shell						NA

[a] The statistically significant results are highlighted

The OO category includes two languages: Java and C++. The p-value equals to 0.011. But C++ only has 10 items, its proportion accounts for less than 5% in this category. It implies that the effect of C++ on the OO category can be ignored.

TABLE VIII. shows the results of the paired-samples test for the SL category. The productivity of shell differs with JavaScript, PHP and python. Likewise, we observed that the productivity of shell also differs with JavaScript, PHP in ISBSG.

The VD category includes SQL, JSP and HTML languages, the p-value of any paired languages is all greater than 0.05, which means the productivity of different languages in the VD category shows no significant difference.

Finding 1.2: regarding the programming languages within each category, the productivity of most languages is not significant different between each other in ISBSG and RDCIOS respectively. But in ISBSG, ABAP and C# show significant differences with other languages within their related categories.

B. *RQ2: What are the relative productivity levels of the four language categories for the software industry and individual organizations respectively?*

We use the mean and coefficients of variation (CV) [4, 5] to reflect relative productivity levels of the four language categories for ISBSG and RDCIOS. To obtain the CV of each category, we divide the standard deviation by the mean. Relative productivity levels are calculated by the following two steps: (I) we set the productivity level of the PO group to 1; (II) given the mean of PO group m, we obtain the productivity level of other groups by dividing their means by m. Higher value indicates higher productivity levels.

In RQ1.2, we found that ABAP and C# show significant differences with other languages within each category in ISBSG. There are 1218 items in the ISBSG dataset that belong to the PO category. Among the 1218 items, the proportion of ABAP items only accounts for 4.7%, which is relatively small compared to other languages. We compare the differences of the means and median between including and excluding ABAP in the PO group. The result shows that such differences are much small. Similarly, the proportion of C# items accounts for 4.58% in the OO group with 1486 items. The items of C# also have little effect on the means and median of the OO group. Therefore, we calculate the means of the PO and OO groups based on the whole dataset.

In RDCIOS, we also found that Shell shows significant differences with JavaScript, PHP and python. The proportion of Shell items accounts for 14.5% in the SL group with 379 items. The mean of the SL group including and excluding Shell are 7.93 and 8.85 respectively. The variation is also small. Therefore, we calculate the mean of the SL group based on the whole dataset. Based on the above analysis, we calculate the population means of ISBSG and RDCIOS based on their whole datasets.

Besides ISBSG and RDCIOS, we also compared our results with the productivity of three related studies: IBM2013, reported by Capers Jones [4], ISBSG2012, reported by Andrew Binstock [25], and ISBSG2016, reported by Luigi Lavazza [5]. Capers Jones reported IBM software productivity of a few popular programming languages based on the data of projects in IBM before 2013. The productivity was measured by function points per month. Base on the ISBSG dataset, Andrew Binstock and L. Lavazza performed an empirical study on the productivity of programming languages in 2012 and 2016 respectively. They also selected a few popular languages.

From the three related studies, we can obtain the productivity of programming languages, measured by function points per time. Then we divided the data into the four language categories, and calculated their productivity levels and CV. Considering the productivity measured by different metrics, we converted the productivity measured by personal hours per function point (PH/FP) to function points per personal hour (FP/PH) to keep a consistent format. As the datasets from existing literatures don't include raw data, only contain average productivity of programming languages, we didn't calculate the CV for those datasets.

TABLE IX. shows the output of the productivity levels and CV in the software industry and individual organizations. ISBSG2012 and ISBSG2016 do not provide data of the SL group. Since the population means of datasets in the software industry and individual organizations are all reasonable, we further explore whether the productivity levels of four categories are significant different by comparing relative productivity levels and CV.

The comparison of relative productivity levels: Fig. 1 shows the comparison of productivity levels between the software industry and the individual organizations. We can observe that: (I) with the evolution of HLPL from the PO to the VD category, the productivity generally becomes more productive in the software industry. However, there is a difference in individual organizations that: the productivity of the SL category is declining. (II) The two individual organizations present a quite similar trend of productivity levels in the last three categories. However, in RDCIOS, the PO category is the highest productive category, but in IBM, the PO category is the lowest one.

TABLE IX. THE SUMMARY REPORT AND PRODUCTIVITY LEVELS FOR THE FOUR LANGUAGE CATEGRORIES

Dataset	**Item**	**PO**	**OO**	**SL**	**VD**
ISBSG	Mean[PH/FP]	16.507	13.645	9.75	9.56
	Converted Mean[FP/PH]	0.061	0.073	0.103	0.105
	Levels	1.00	1.21	1.69	1.73
	StDev	15.954	14.548	8.32	10.58
	CV	96.65%	106.62%	85.33%	110.67%
RDCIOS	Mean[Loc/PH]	13.11	12.55	8.85	10.57
	Levels	1.00	0.96	0.68	0.81
	StDev	23.41	15.48	17.60	12.55
	CV	178.57%	123.40%	198.81%	118.74%
IBM 2013	Size	7	10	6	5
	Mean[FP/PM]	7.273	11.50	9.27	11.26
	Levels	1.00	1.58	1.27	1.55
ISBSG 2012	Size	4	4	NA	2
	Mean[PH/FP]	15.98	11.75	NA	8.45
	Converted Mean[FP/PH]	0.063	0.085	NA	0.118
	Levels	1.00	1.36	NA	1.89
ISBSG 2016	Size	3	4	NA	2
	Mean[FP/PM]	18.53	19.35	NA	25.85]
	Levels	1.00	1.04	NA	0.94

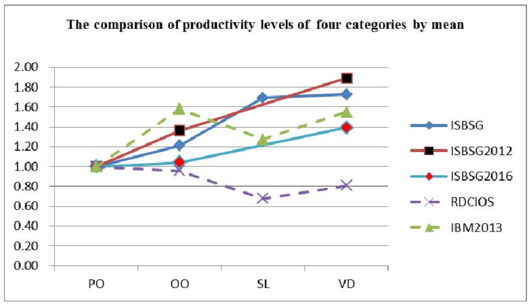

Fig. 1. The comparison of productivity levels of thefour language categories

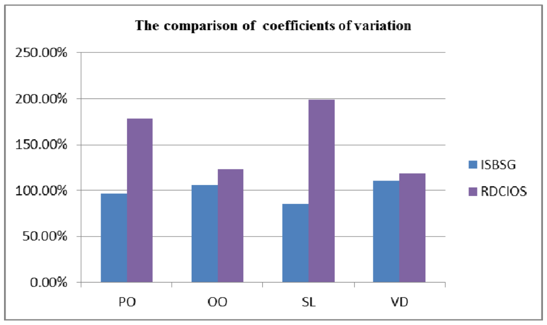

Fig. 2. The CV comparison of the four language categories

The comparison of CV: Fig. 2 shows the comparison of CV between ISBSG and the organization RDCIOS. We can observe that RDCIOS has higher CV than ISBSG, which indicates that data size in ISBSG is much more concentrative and the dispersion is small, while the RDCIOS data looks more disperse. The reason is that continuous process improvement has improved its productivity since 2017.

Finding 2: The results strongly indicate that: in the scope of the software industry, the relative productivity levels are gradually increasing with the evolution of HLPL. However, the individual organizations don't always keep the same increasing trend. For both individual organizations, the productivity of SL category is declining, and they have similar trends of relative productivity levels except for the PO category. In addition, the CV of RDCIOS shows larger than ISBSG, especially in the PO and SL categories.

C. RQ3: Regarding an individual organization, can the productivity performance baseline be established based on the four categories to support statistical process control?

Many software organizations conduct effort estimation based on their productivities. Particularly, a CMMI-adopted organization needs to establish the process performance baseline to support quantitative management. In RQ1 and RQ2, we observed that individual differences existed between the software industry and the individual organization both in productivity differences and relative productivity levels across the four language categories. It indicates that software organizations need to establish the productivity baselines based on their own data, rather than simply adopt the industry baseline. Proper productivity baselines can support the statistical process control. In RQ3, we aim to illustrate whether the individual organization can establish its productivity baseline based on the four language categories.

We applied mean and standard deviation charts (Xbar-S) to establish the Statistical Process Control (SPC) of personal average productivity. The process of constructing dataset for the Xbar-S charts includes: (I) we considered lines of code per week as the productivity metric. We obtained the individual weekly productivity for each category. (II) Most projects in RDCIOS adopted the agile development model, and took one month as a milestone, which included requirement analysis, design, coding and test. Therefore, we calculated the average productivity of each successive four weeks as a control point. Each control point denotes the productivity per month, measured by lines of code per week. The Xbar-S charts of the four language categories are shown Fig. 3.

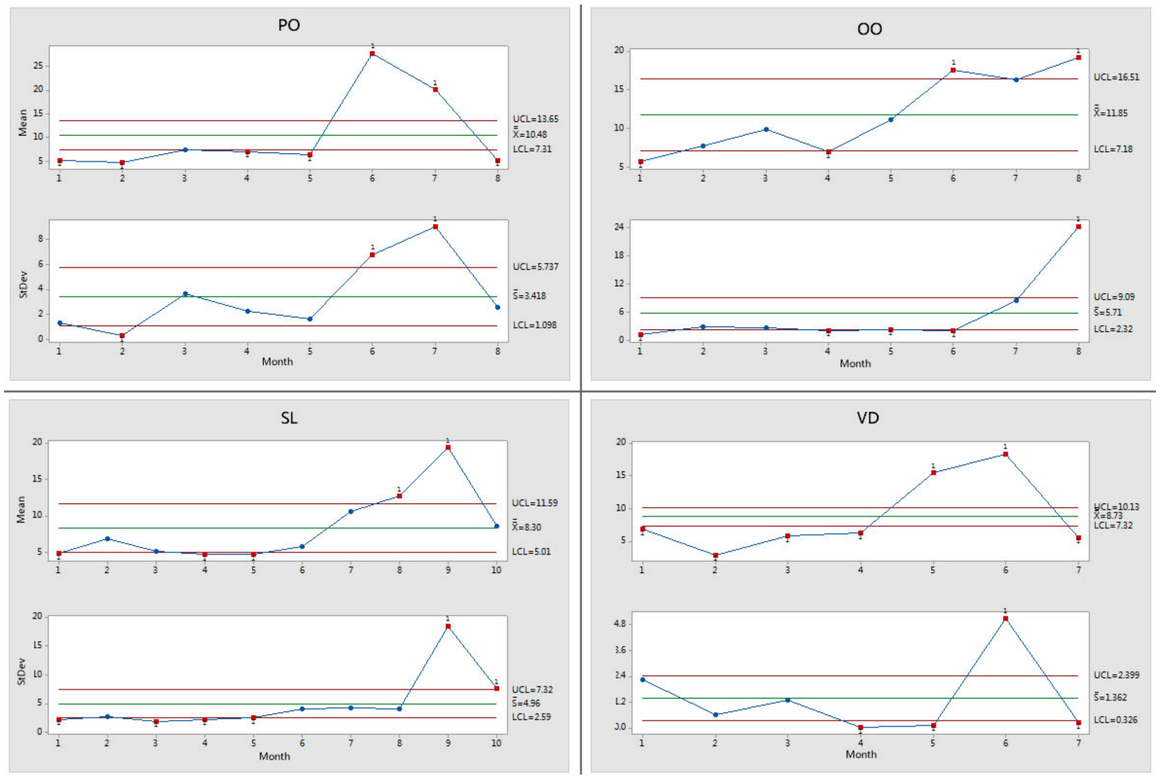

Fig. 3. The Xbar-S Charts of the four language categories in RDCIOS

We can observe that: (I) the productivity fluctuations of the four language categories are balanced in the first several months, which means that the control mean of each language category is statistically meaningful. Therefore, the individual organization can generate its productivity baseline by using the average productivity. (II) the productivity has improved obviously in the last few months. The baseline may be updated if the upward trend is kept.

Finding 3: The statistical process control shows that the programming productivity of the four language categories is mostly stable. The average productivity of the four language categories can be used to establish a productivity baseline for the investigated organization.

IV. Threats to Validity

Threat to internal validity lies on experimental bias. In ISBSG, the highest quality data and project size measured by function points, are rated by 'A', followed by 'B','C' and 'D'. To keep the data in a high quality, we selected the data rated by 'A' or 'B, and quality of project size rated by 'A' or 'B'. In RDCIOS, we measured productivity of the organization based on the lines of code, without considering the documents, which may not reflect the productivity comprehensively. Therefore, we excluded the commits that mainly contain documents, and quite few codes.

Threat to external validity is related to the generalizability of our findings. In this work, we analyzed productivity of different languages in software industry ISBSG and the individual organization RDCIOS. Although we collected data from real-world datasets, the findings may not be generalizable to all the other software organizations. The more stable the process is, the more general the productivity baseline becomes.

V. Related Work

Literatures related to the productivity of programming languages mainly focus on two aspects: (I) the metrics of productivity. (II) The variations across different languages.

A. *The Metrics of Productivity*

Different researchers used different metrics to measure productivity. Petersen [13] performed a systematic review on the measurements and predictions of software development productivity, which focused on quantifiable approaches, such as metric spaces and data envelopment analysis. In [11] authors investigated productivity metrics in agile software development, including source lines of code per time, function points tasks completed per time. Other approaches have been proposed, including productivity metrics based on different development roles [14], productivity measure by resolved issues per month [12]. In summary, lines of code and function points are largely used when measuring productivity. Function points work well when measuring productivity for projects that requirements are well documented and stable [21].

B. *The variations across different languages*

Previous studies investigated the variations across different languages mainly based on three aspects: productivity, development and maintenance. Premraj et al. proved that the choice of the most appropriate programming language could affect their productivity and final development costs [6]. Sebastian Nanz et al. [17] performed an empirical study to investigate which language makes developers more productive; Myrtveit et al. investigated whether development in C++ was more productive than development in C [20]; In [3] authors explored the lines of code of different languages per function point. Similarly, C. Jones [2] analyzed the productivity levels of different languages, which was enriched in 2013 to refine further productivity of the languages based on data of IBM projects [4].

Another two studies [25, 5] analyzed productivity levels based on the ISBSG dataset in 2012 and 2016 respectively, which are very similar to our study, but only analyzed a few popular languages. In addition, a few studies [19, 21, 22] investigated whether programming languages affected productivity in Open Source projects. For the development and maintenance variations, some studies [1, 15, 16] compared the development performances of different languages from various aspects: program length, runtime efficiency, etc. Other studies focused on the challenging and pressures when the programming languages upgraded [23], quantifying the impact of programming languages on software development and maintenance 18], and investigating the impact of programming languages choices on the maintainability of OSS projects [24].

Some studies show that the productivity varies among different programming languages. But little research work has been done to consider the programming language categories and comprehensively analyze the productivity variations across different languages. Our work fills that gap.

VI. Conclusion and Futrue Work

In this paper, we proposed a classification to divide HLPL into four categories based on their evolution, and revisited their productivity variations in ISBSG and RDCIOS. The results show that: (I) the programming productivity varies across the four categories in ISBSG and RDCIOS, which shows that the four categories of HLPL are reasonable. (II) Individual differences exist between ISBSG and RDCIOS. In ISBSG, the relative productivity levels are gradually increasing with the evolution of HLPL. However, in RDCIOS, it does not always keep the same increasing trend. (III) We present an example to illustrate that the productivity baseline can be established based on the four categories for RDCIOS and the statistical process control can be applied based on its baseline. These results can help managers better understand the programming productivity variations across different languages, and guide the software organizations to establish their productivity baselines, rather than simply adopt the industry baseline. The established baselines can support the quantitative management for organizations.

In RDCIOS, we analyzed the productivity only based on submitted codes. In the future, we plan to investigate the submitted document-products at the same time. In addition, we also plan to establish the relationship between submitted products and the tasks created in JIRA to analyze the productivity of different types of issues.

Acknowledgment

We appreciate anonymous reviewers for their supportive and constructive comments. This research is supported by

National Natural Science Foundation of China under Grant Nos. 61432001.

References

[1] Prechelt, “An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program”[J]. IEEE Computer, 2002.

[2] Jones, C. “Programming Languages Table”, Software Productivity Research, Inc. Release 8.2. March 1996.

[3] Marketwired, “Quantitative Software Management, Inc. Function Point Languages Table 5.0”. April 2013. http://www.qsm.com/resources/function-point-languages-table

[4] Jones, C. “Function points as a universal software metric”. ACM SIGSOFT Software Engineering Notes 38.4, 2013.

[5] Luigi Lavazza, Sandro Morasca, Davide Tosi, “An empirical study on the effect of programming languages on productivity”, Proceedings of the 31st Annual ACM Symposium on Applied Computing, 2016. pp: 1434-1439.

[6] Premraj, R., Shepperd, M.J., Kitchenham, B.A. and Forselius, P. 2005. “An Empirical Analysis of Software Productivity over Time”, 11th IEEE International Software Metrics Symposium, METRICS 2005.

[7] Programming Languages, “The University of TOLEDO college of Engineering”. http://cset.sp.utoledo.edu/sample/engt1050/engt1050_languages.html

[8] “The different types of languages”, http://landofcode.com/programming-intro/computer-programming-languages.php.

[9] Ken Bigelow, “Levels of Programming Languages”, http://www.play-hookey.com/computers/language_levels.html

[10] Wikipedia, the free encyclopedia, “High-level programming language”. https://en.wikipedia.org/wiki/High-level_programming_language

[11] Syed Muhammad Ali Shah, Efi Papatheocharous, Jaana Nyfjord, “Measuring productivity in agile software development process: a scoping study”, International Conference on Software and System Process, 2015.

[12] Athanasiou, D.; Nugroho, A.; Visser, J.; Zaidman, A., “Test Code Quality and Its Relation to Issue Handling Performance”, Software Engineering, IEEE Transactions on , 2014, vol.40, no.11, pp.1100-1125

[13] Petersen, K. “Measuring and predicting software productivity: A systematic map and review”. Information and Software Technology. 53, 4 (2011), pp:317–343.

[14] Hernández-López, A., Colomo-Palacios, R. and García-Crespo, Á.. “Software Engineering Job Productivity—a Systematic Review”. International Journal of Software Engineering and Knowledge Engineering, 2013, pp:387–406.

[15] L. Prechelt, “An empirical comparison of seven programming languages”, IEEE Computer, 2000, vol. 33, no. 10, pp. 23-29..

[16] L. A. Meyerovich , A. S. Rabkin, “Empirical analysis of programming language adoption,” International Conference on Object Oriented Programming Systems Languages & Applications, 2013, pp. 1–18.

[17] S. Nanz and C. A. Furia. “A comparative study of programming languages in rosetta code”. In Proceedings of the 37th International Conference on Software Engineering - Volume 1, 2015, pages 778– 788.

[18] P. Bhattacharya, I. Neamtiu, “Assessing programming language impact on development and maintenance: A study on C and C++”, International Conference on Software Engineering, 2011, pp. 171-180.

[19] T. E. Bissyandé, F. Thung, D. Lo, L. Jiang, L. Réveillère, “Popularity interoperability and impact of programming languages in 100000 open source projects”, Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference, 2013, pp. 303-312.

[20] I. Myrtveit and E. Stensrud. “An empirical study of software development productivity in C and C++”. In NIK'08.

[21] J. Krein, A. MacLean, D. Delorey, C. knutson, and D. Eggett. “Impact of programming language fragmentation on developer productivity: a source forge empirical study”. International Journal of Open Source Software and Processes., 2010, pp:41–61.

[22] P. Delorey, Charles D. Knutson, and Scott Chun. “Do Programming Languages Affect Productivity? A Case Study Using Data from Open Source Projects”. In 1st International Workshop on Emerging Trends in FLOSS Research and Development, 2007.

[23] Urma R G, Orchard D, “Mycroft A. Programming language evolution workshop report” [C]// The Workshop on Programming Language Evolution. ACM, 2014:1-3.

[24] Celia Chen, Lin Shi, Kamonphop Srisopha, “Maintainability Index Variation Among PHP, Java, and Python Open Source Software Projects”, In Proceedings, the 27th Annual IEEE Software Technology Conference(STC), 2015.

[25] Andrew Binstock, Peter Hill, The Comparative Productivity of Programming Languages, 2012. http://www.drdobbs.com/jvm/the-comparative-productivity-of-programm/240005881