International Journal of Software Engineering and Knowledge Engineering

World Scientific
www.worldscientific.com

SOFTWARE ENGINEERING JOB PRODUCTIVITY - A SYSTEMATIC REVIEW

ADRIÁN HERNÁNDEZ-LÓPEZ RICARDO COLOMO-PALACIOS
ÁNGEL GARCÍA-CRESPO

Computer Science Department, University Carlos III of Madrid, Av. Universidad 30, Leganés, Madrid 28911, Spain
adrianhernandezlopez@gmail.com
{ricardo.colomo, angel.garcia}@uc3m.es

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Productivity is a key element in organizational management. Although it can be measured at different levels (country, sector, organization ...) this research focuses on productivity at the job level. The aim of this paper is to obtain an overview of the state of the art in productivity measurement in software engineering, including the inputs and outputs of the production process used for this measurement at the job level in the workplace. To do so, a systematic literature review protocol was adapted from literature review protocol standards, and subsequently carried out. The objective is to assess the current inputs and outputs present in the literature in order to create new productivity measures for software practitioners. This paper reveals that two different measures are used to assess software engineering professionals: traditional SLOC/Time and planning projects units per time unit.

Keywords: software engineering; job productivity; systematic review; software economics.

1. Introduction

Productivity management remains a challenge in Information and Communication Technology (ICT) project management [1]. While in the manufacturing industry there already exist designed and tested methods for determining productivity, the ICT industry is one step behind in terms of available methods for evaluating the outputs and predicting the effort needed to complete projects [2]. The difficulty of managing productivity in this sector is due to a high dependence on human resources in its activities [3]. It can be said that the productivity of personnel is one of the components that contributes to software quality [4]. Within the ICT industry, Software Engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software [5].

The measure of productivity in software projects is mainly based on ratios between the size of the delivered software and the effort to obtain it [6]. Thus, the most frequent methods of measuring the size of the delivered software are Function Points (FP) and

Source Lines of Code (SLOC). However, these measures are of questionable reliability and the overall development can be unproductive even if productivity grows at the development level [7]. There are also more recent efforts, such as measuring post-mortem FP [8], multiple size measurement [9], and specific methods for applying concrete methodologies such as Object Orientation (OO) [10]. Additionally it is noteworthy that the measurement of productivity based on SLOC only represents programming activities, while other activities such as management, analysis and design are beyond the scope of these measures [11]. Similarly, the FP based measures represent only the functionality offered by the developed software, but ignores non-functional development, or other elements such as reuse or accessibility.

Factors affecting ICT productivity are accepted and recognized by most organizations: time constraints, reliability requirements, high-level coding languages, the size of the development team, the volatility of requirements, staff skills in using programming tools, customer involvement, and duration of the project (see for example [12]). Despite their acceptance and recognition, the influence of some of these factors, both positive or negative, in ICT productivity are not clear and may vary depending on external factors such as the business sector [12] and outsourcing ratio [13].

In this scenario, the definition of new models, measures and evaluation methods can shed light on productivity in ICT projects, especially in software development projects similar to that created for software maintenance projects [14]. In turn, organizations need to establish their own measure of productivity in addition to existing measures in the sector in order to carry out benchmarking of their data [12]. Before defining new models, measures and methods to measure productivity in software development projects, the definition of the state of the art is needed to provide an overview of the current situation. Therefore, this paper presents the state of the art in the measurement of productivity at the job level in SE.

The remainder of this paper is organized as follows: first there is a summary of the measurement of productivity in SE; second, the questions to be answered by the systematic review are presented; third, the planned Systematic Literature Review (SLR) process is outlined; fourth, the results of the execution of the SLR are presented; fifth, main findings are presented; and finally, there is a discussion of the results and findings.

2. State of the Art

The origins of the term “productivity” go back to the eighteenth century, when Quesnay introduced it [15]. Traditionally, productivity has been defined as the ratio of output units produced per unit of input [16]. Although this definition fits well in manufacturing paradigms because the output is based on quantities of standardized and clearly identified units of measurement, the definition does not adequately apply to new environments such as service industries, where intangible assets are also produced. While manufacturing industries have existing designed and tested methods for determining productivity, the IT industry lags behind in terms of methods for evaluating outcomes and predicting the effort required to complete projects [1]. Productivity management is still a

challenge for the software industry [17] and continues to be under study even in new aspects of programming techniques like extreme programming without establishing a clear result [18]. The traditional definition of productivity, the ratio of output units produced per unit of input effort, is not easily applicable to software development, since labor costs are by far the largest production expenditure [19]. The literature has suggested several factors that influence productivity within this context: timing constraints, reliability requirements, languages skills, team size, requirements volatility, staff skills in using programming tools, staff availability, customer participation, and project duration [12]. However, their influence on productivity is not clear and other external factors such as the business sector [12] and outsourcing ratio [13] may come into effect.

The IEEE Std. 1045-1992 defines productivity as the relationship of an output primitive (source statements, function points or documents) and its corresponding input primitive (effort, e.g. staff-hours) to develop software. In this direction, and according to Koch [20], productivity in software development considers the relation of output and effort and is often assessed by using SLOC, FP [21] or further revisions of FP like post-mortem FP [8]. A FP can be defined as a unit of measurement to express the amount of business functionality an information system provides to a user. FP is usually preferred over SLOC because of the independence of the programming language. These approaches may be problematic due to the missing components, especially from the output point of view and, thus, productivity measures of software development need to contemplate several different types of measures. In spite of their controversial use, these two measures for measuring output continue to be used [22] when using classical development methods, but alternatives such as story points for agile development have recently appeared [23]. Below are some examples of productivity measures used in SE:

(a) Productivity = (Output / Input) * Factors
(b) Productivity = SLOC / t, where t is a unit of time or effort (e.g. man-month or hours)
(c) Productivity = FP / t, where t is a unit of time or effort (e.g. man-month or hours)

Moreover, software development comprises a set of tasks influenced by products, processes, methods, techniques, technologies, tools, and people. In addition, software development requires a high level of cognition skills in software engineers which further increases its complexity [24]. This variety of factors implies a high level of complexity for software development tasks, including construction and testing among others [25]. Moreover, studying software engineering is complex not just because of its technical aspects, but also "from the awkward intersection of machine and human capabilities, and from the central role of human behaviour in software development" [26]. The complexity of software development implies that productivity is difficult to measure [27]. But as Anselmo and Ledgard stated [28], following the statement of Lord Kelvin - *"When you can measure what you are speaking about, … you know something about it; but when you cannot measure it, … your knowledge is of a meager and unsatisfactory kind…"*, so we cannot expect to improve software productivity without measuring it. Thus, an appropriate productivity measure provides a forecasting tool to achieve this improvement in productivity [29]. Furthermore, managing software projects is complex, since managers need to deal with personnel, team and organizational resources [30]. In this

sense, assessing the productivity and efficiency at the job level is crucial. Effective productivity management in software projects requires many factors to be considered. There are thousands of possible factors and considering all of them is not feasible from a managerial point of view. Therefore, productivity modelling should focus on the limited number of factors which have the most significant impact on productivity [31].

In SE, the scientific focus on productivity measurement can be said to have begun in the late 80's. Originally, the focus was primarily programming activities [32] and software projects [33]. In addition, at that time some authors began to consider the factors that affect productivity [34-35]. Over time, and considering that one objective of the measurement of productivity is the comparison of measures, some studies focused on that aspect [36-37]. Similarly, the study of factors continued in the following decades [38]. On the other hand, some elements such as reuse [39-40] or OO [41] started to be included in the measurement of productivity. However, the use of these elements appears to have not been widely accepted, since 80's methods continued to be used [42-44]. The underlying purpose of these measurements is to measure developed lines of code (SLOC) or the functionality delivered (FP). These measurements are also used for future software project estimation applying any of the available methods [45].

In this scenario, given that one of the classic challenges of the software industry is the timely delivery of projects [46], the use of productivity measures that aim to measure the efficiency of delivery should not be surprising [47]. Nevertheless, these productivity measures do not measure all the activities of SE, and therefore do not measure productivity at the job level. Thus, it is necessary to develop measures of productivity for that level.

Before any attempt to measure productivity is made, it is necessary to identify what should be included in this measurement [48]. Thus, taking into account these contributions, and with the aim of creating a measure of productivity in SE, it is necessary to distinguish the factors (1), inputs (2) and outputs (3) that can constitute measures.

(1) **Factors** affecting productivity can be classified into two main areas: technical factors (product, process, development environment) and soft factors (organizational culture, team culture, skills and experience, environment, project) [49]. Although many of these factors are widely known and are also used in estimation models such as COCOMO [50] - for example, the influence of the experience of workers and the programming language (development environment) [51-52] - it is still unclear that its importance is not what it once was because practices, work processes and tools have evolved considerably since initial studies (for example [51]). In addition, the evaluation of the degree of influence and valence of the factors is a difficult task. This difficulty is justified because the influence of factors is not always the same and will depend largely on the characteristics of the project and the work environment [53]. What does seem clear is the existence of two groups of factors: personnel factors and technological factors [49].

(2) **Inputs** of the production process that have been measured and used so far focus on units of time [54]. Although the distribution of time should not be homogenous [55],

this is a crucial factor in regard to input in productivity. The use of time, as the only input, may be due to two concerns [56]:

a. the delivery of a project at the scheduled time and the placing on the market of the same elements are important factors for a successful project, and
b. the biggest cost in software producing organizations is related to the effort of the staff, so the time dedicated is the main cost, i.e. more time producing the software implies more cost due to the payment of software engineers.

Both interests in using this input are economic, but productivity is not only an economic indicator. Moreover, the use of this input leaves some questions unanswered: Is time the only input needed to produce software? And if time is the only measure of input: should we measure the time used by the employee to carry out their tasks or the time paid for by the employer? In parallel and with the advance of time, hardware equipment costs have been reduced and their cost is almost irrelevant in large projects. Thus, any resource of real interest in the measurement of productivity is not related to this particular resource, but will be directly related to the workforce. Time is, without doubt, one of them when the use of labor is increased, however, time does not seem to be the only input while still thinking of SE as a craft activity [46].

(3) **Outputs** of the production process have followed the same philosophy of measuring the time used to deliver the software project; therefore outputs have been measured primarily in source code (SLOC) or functionality (FP) [21]:

a. FP is a measure of the functionality to be developed and, therefore, a way to measure the functionality delivered to the client considering functionality as the only output.
b. SLOC are the instructions that encode the functionality developed, so are alow level measurement making them more tangible. Nevertheless, these two types of output are not the only ones that occur in SE.

These measures of output, despite their limitations, are popular in organizations [8] and research scenarios alike. But there is a widely accepted output which is beyond those usually considered: quality. Quality affects the output produced (effectiveness) and the production process itself (efficiency) [57]. In addition, there are features of the outputs that can affect productivity, both for its initial creation and its subsequent use, for example reuse [28] and documentation [58].

Additionally, the level of analysis should be taken into account when measuring productivity in SE, since not all measures are useful at all levels [43]. The purpose of the measures and the audiences interested in the results of these measures must be clearly specified; this information must be defined before any other information on productivity [59]. At the sector and organization level, the granularity of the factors, inputs and outputs involved is not sufficient for accurate measurement that can be used as a source of information for improving productivity; so the measures at these levels are only one

element for comparison between periods. At lower levels of measurement, a comprehensive analysis of the elements to be measured is required, which calls for an established process (see for example [59]). At this point, it is necessary to ask whether current practices for measuring SE productivity follow this process, and if the factors, inputs and outputs are clearly identified at each of the levels of measurement; and if widely accepted factors such as reuse or quality are included (directly or indirectly) in the measures of productivity used.

3. Systematic Review Protocol

One way to construct an overview of state of the art is by using the type of study called Systematic Literature Review (SLR). A SLR is a means of identification, evaluation and interpretation of all available research relevant to a specific research question or a subject area or phenomenon of interest. Although currently there are several references that explain the process of designing and implementing a SLR, for the purpose of this study the authors chose the reference manual adapted from Kitchenham [60]. After reviewing the literature on SLR for similar research objectives, it can be stated that there is no previously published search strategy for a systematic review around the measurement of productivity in SE at the job level. Specifically, three existing systematic reviews on productivity in SE have been found, but their purpose is not the same: two of them review the factors affecting productivity [49, 61] and the other one is related to the methods used to measure and to predict productivity at the project level [45]. On the other hand, no SLR has found the measurement of productivity at the specific job description level within jobs that require advanced technical knowledge and skills. Although there is a lack of previous SLR with similar objectives, there are other investigative projects lacking methodology, such as a taxonomy of productivity measurement [3]. Thus, the search strategy presented below has been designed and adapted from the previously found SLR .

3.1. *Research questions*

This SLR aims to summarize and clarify the inputs and outputs measured and used in productivity measurement at the job level in SE. Toward this aim, two research questions (RQs) were raised as follows:

(1) RQ1. What are the inputs and outputs of the software engineering process?
(2) RQ2. Are there different inputs and outputs for different SE job roles?

In addition to these RQs, if the answer to RQ2 is affirmative, then the following RQs should be answered:

(3) RQ2.1. What are the inputs and outputs for the software engineer?
(4) RQ2.2. What are the inputs and outputs for the programmer?
(5) RQ2.3. What are the inputs and outputs for the analyst?
(6) RQ2.4. What are the inputs and outputs for the project manager?

These questions are added to determine the inputs and outputs used for each of these software engineering jobs. If there is an affirmative answer to RQ2 and some of these

questions are answered, it will be possible to compare the inputs and outputs used for each of these jobs and propose a specific and different measure of productivity for each one of them. On the other hand, if the answer to RQ2 is negative then it would not be possible to answer questions RQ2.1 to RQ2.4.

There are no questions related to factors, in spite of having mentioned them in Section 2. This decision has been made considering that (1) there is some research related to factors (see for example [11, 49, 61]), and (2) factors are not generally included in productivity measures, they are taken into account when measuring some of the parts, for example when measuring FP.

3.2. *Search strategy*

The search strategy comprises search terms, literature resources and search process, which are detailed one by one as follows:

3.2.1. *Search terms*

The search string has to be defined based on the population under study, and the keywords and their synonyms. Therefore, the study population includes the inputs (e.g. man-hours, knowledge...) and the outputs (e.g. SLOC, FP, requirements, design artifacts, documentation...) of the jobs related to the development software process of the SE (e.g. software engineer, programmer, project manager, analyst, designer...). With this population the list of keywords and their synonyms, used to generate the search string was:

- Productivity: performance.
- Input: resource.
- Output: product, service.
- Personnel: staff.
- Software engineering: software development, software maintenance.

It should be noted that the selected synonymous of SE are the two main areas that account for much of the activity and in which the jobs under study have direct influence. To generate the search string a Boolean language with AND and OR, and quotation marks for exact text were used. The string format is recognized by all sources of information used, as well as many others. So finally the search string used is as follows: *("software engineering" OR "software development" OR "software maintenance") AND (productivity OR performance) AND (input OR resource OR output OR product OR service) AND (personnel OR staff)*

3.2.2. *Literature resources*

Given the diversity of sources to be consulted electronically via the web, six electronic databases of established literature resources were used for the present SLR (*IEEEXplore, ScienceDirect, SEI Web of Science (WoS), Wiley Online, ACM Digital Library, Taylor & Francis*).

3.2.3. *Search process*

The procedure for selecting studies was as follows: first, the search string was executed in the search engine of each selected source of information; secondly, to select an initial set of studies, abstracts of all retrieved studies were read and evaluated according to inclusion and exclusion criteria, and thirdly, to refine the initial set of studies, each full article was retrieved and read to verify their inclusion or exclusion. The reason for exclusion or inclusion in this third stage was documented; fourth, once these primary studies were selected, their references were analyzed to identify any studies that had not been found in the initial search and which should have been analyzed. This procedure (see Figure 1) was based on that used in a recent SLR [62].

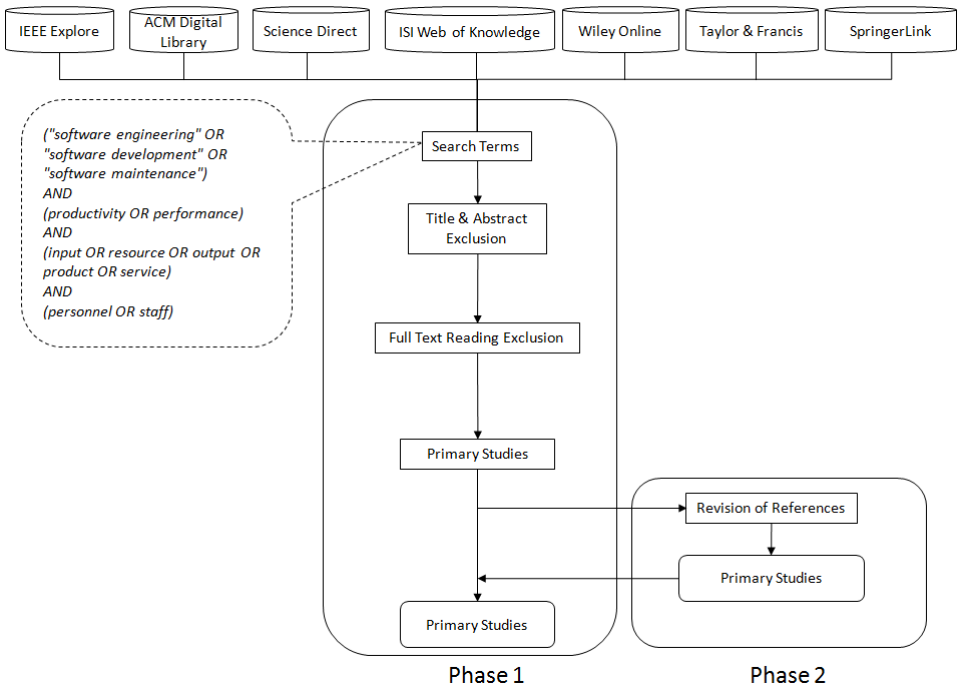

Fig. 1. Filtering process based on [62]

3.3. *Study selection*

Once initial search results were retrieved, an exclusion/inclusion review procedure was applied with the following inclusion and exclusion criteria:

- Inclusion criteria
  - Papers that have empirical content.
  - Papers on measuring productivity in SE.
- Exclusion criteria
  - Papers that do not have empirical content.
  - Papers that include only factors and are not about input or outputs.

- Papers that return the same sample data.
- Papers that are not related to SE.
- Papers that measure higher levels of productivity (e.g. team productivity, Project productivity, department productivity…) and do not take into account the individual level.

3.4. *Data extraction*

The entire search process was documented and items were stored in a reference manager. In this tool, groups of elements based on the result obtained in each filter were created. And for items that passed the first filter, the full text was recovered, which was also stored in the reference manager. For the final elements, a form with the necessary data for further analysis was completed for each one. With the information collected in that form, it was possible to obtain qualitative and quantitative information to answer the planned research questions. In particular, the following information was collected:

- Types of inputs
- Types of outputs
- Types of inputs by each job
- Type of outputs by each job
- Measure of productivity used
- Measure of productivity used by each job

4. Systematic Review Execution

Included and excluded studies are presented in stages following the search process described above. Given the length of some of the list of references, they have been hosted online and can be downloaded at any time.

4.1. *Primary studies obtained in the first phase*

The first phase of the SLR search process was executed on 16 May 2011. The result was as shown in Table 1.

Table 1. First phase results without filtering

IEEExplore	110
ACM Digital Library	5
ScienceDirect	11
SrpingerLink	0
Willey Online	38
ISI WoK	13
Taylor & Francis	0
Total	187
Total (without duplication)	177[a]

[a] http://dl.dropbox.com/u/20836838/Lista_Inicial.pdf

Of the 187 results obtained, ten were duplicated; in particular, these duplicates were obtained from ISI WoK. Of these remaining 177 results, 22 were discarded for being incomplete or not related to SE. Of the 155 remaining, 98 were excluded after reading the title and abstract, so 120 results were excluded in the first filter, which left 51 results to be filtered by full-text reading using the inclusion and exclusion criteria. Of the 51 useful results for the next filter, 27 were journal papers and 24 were conferences papers. After the first filtering process, if each result did not meet the exclusion criteria after reading the full text they were included. The results are shown in Table 2. The result of the first phase was that three primary studies were obtained. These results consist of a journal article and two conference papers.

Table 2. First phase results

Excluded	48
Included	3
Total	51[b]

4.2. *Primary studies obtained from the second phase*

The reference lists from the primary studies obtained from the first phase were retrieved and the same filters previously used were applied to them. A total of nine references were obtained by reading the title and abstract. From these references, three were finally selected using the criteria of inclusion and exclusion. Thus, these results raised a total of six primary studies as shown in Table 3.

Table 3. Second phase results

First phase results	3
Second phase results	3
Total	*6*[c]

5. Results and Findings

5.1. *Results*

Table 4 shows the results of the completed SLR. Six primary studies were obtained ($N = 6$); four are conference papers and two are journal papers. Table 5 shows the details of each result. In some of the results there were some measures of inputs and outputs not used in the measurement of productivity, but that have been included in Table 5 because they measure parts of productivity individually.

Table 4. Quantitative results; number in brackets represents the number of occurrences

[b] http://dl.dropbox.com/u/20836838/Lista_Incluidos-Excluidos.pdf
[c] http://dl.dropbox.com/u/20836838/Lista_Incluidos-Excluidos_Fase2.pdf

Input	Time (5)	Assigned tasks (1)	
Output	LOC (4)	Completed tasks (2)	Rate (difficulty and effort) (1)
Measure of productivity	LOC/Time (3)	Tasks/Time (3)	
Data simple type	University (3)	Organizational (2)	Simulation[d] (1)

Table 5. Studies details

Phase	Study	Measure	Input	Output
1	[63]	Milestones/m	Minutes (m)	Milestones
1	[64]	NCSS/h	Hours (h)	Noncommentary Source Statement (NCSS)
1	[65]	(Completed program)/h	Hours (h)	Completed program
2	[66]	(Completed tasks)/h	Hours (h) Assigned tasks	Completed tasks Rate (difficulty and effort)
2	[67]	SLOC/h	Hours (h)	SLOC
2	[68]	SLOC/h	Hours (h)	SLOC

Thus, it is now possible to answer the research questions posed:

RQ1. *What are the inputs and outputs of the software engineering process?*

Inputs are: a) units of time (mainly working hours and days); b) tasks assigned to workers; while outputs are lines of code and completed tasks.

RQ2. *Are there different inputs and outputs for various jobs in SE?*

This question could not be answered with certainty because in all of the primary results obtained, productivity is measured without a clear job definition. Since this question cannot be answered with certainty, the authors are unable to answer questions RQ2.1 to RQ2.4.

5.2. *Findings*

With regard to inputs, there is a clear tendency to use time measures (hours, days, weeks...) but also one of the results ([66]) uses the assigned tasks as a measure for input. In this study, the authors recommended the use of the same tasks for all participants in the study, i.e., all participants performed the same tasks. This approach is virtually impossible to implement in a real SE working environment, because each task is unique and is performed, except in the case of re-work, just once. But it is also worth noting that this input is more complex and, as a result of this, could be a better input for the measurement of productivity at the specific job description level. In any case, standard time measures (hours, days, weeks…) are more important according to the importance of their references in the literature.

[d] Two scenarios were simulated to compare between pair programming and classical development and some parameters were fixed for both options.

With reference to outputs, this aspect presents two results of equal importance: SLOC and completed tasks. In one of the results ([66]), tasks are additionally assessed by the difficulty and effort needed to complete each task. This finding raises an alternative to productivity measures based on the size of the output (source code or functionality). Specifically, this alternative allows the use of each task assigned and completed as an output of the production process of each worker. On the other hand, the use of a measure of size, such as the SLOC, is not surprising since it is widely used when measuring the productivity at different levels and is even used for measures related to estimation. This measure is easy to take, but there is no common measurement for it. However, according to DeMarco [69], the problem of using SLOC is not the lack of a common measurement, but its late measurement. In addition, various SE tasks (e.g. requirements specification or design) are hardly convertible into SLOC, or other size measures.

Considering the presented results for the inputs and outputs, the results for the formula used to measure productivity can only be combinations of them. Thus, results are divided into two groups: LOC/Time and Task/Time. It should be noted that although a primary study defined the task assignments as an input, it is only used to measure productivity as a unit of time. So, from the results it could be said that all the inputs used in the measures of productivity are any type of measurement of time. Importantly, all found productivity measures are based on the visibility of worker efficiency during the contracted period which limits the visibility of the measurement of productivity to just efficiency [70]. Furthermore, these measures cannot be customized in order to consider each organizational and cultural need and each work environment, when and where they arise. Moreover, these measures are partial productivity measures as they don't have to take into account all the inputs or outputs [71]. Thus, none of the primary studies deals with the need to have a total measure of productivity per job that can be adapted to each culture, organization and job. Finally, with respect to the types of data sources used in each of the primary studies, the results are in line with the trend in research of using university participants to perform the studies, followed by the use of real workers [72].

It should be noted that there is no clear differentiation of jobs in any of the studies obtained. So, considering the outputs used in these studies, only the productivity of jobs focused on programming (SLOC) could be measured with this kind of measure. On the other hand, measures based on completed tasks are more general and can be applied to countless jobs where there is a clearly identifiable task-work structure.

In the second phase, three results were discarded during the reading of the full text. These results were reference books widely used in the SE area ([53, 69, 73]). Although these results were discarded, all include references to productivity measurement and to the factors that affect it. For instance, Sommerville [53] gives the following definition of productivity: *"Software productivity is an estimate of the average amount of development work that software engineers complete in a week or a month. It is expressed as lines of code/month, function points/month, etc."* This definition is in line with the presented results of the SLR because it uses a measure of product for measuring the outputs, a measure of time for the inputs and a ratio formula for measuring productivity. In addition,

Sommerville [53] discusses the difficulty of measuring productivity considering that the inputs and outputs are not entirely tangible, to which he adds the influence of the quality of the production process with the inherent difficulty of measuring impact on productivity.

On the other hand, DeMarco [69] indicated that differences between workers' productivity can be enormous, as was previously suggested by other investigators [74-76], a difference that can be increased if the team size increases. In addition, DeMarco [69] suggested that time and people, as inputs into the production process, are not interchangeable in productivity models although there is some degree of exchange between these two inputs. In addition, he added a rule to improve productivity in software development based on the results of previous studies: *"Taking a poor performer off your team can often be more productive than adding a good one."* Once again, personnel issues and differences in productivity are key issues to understand productivity phenomenon in the software arena.

6. Discussion

Results suggest that there is some unanimity on the inputs used in measuring productivity at the specific job description level. This unanimity is a reflection of the inputs used to higher levels of measurement, for example at the organizational level, in which the input used is (almost) always a unit of time or effort (see for example [9, 28]). The only use of time at lower levels where the level of detail may be higher, would fit perfectly into measures of productivity for manufacturing work, where each task and operation have a defined time and therefore it is planned and measured based on these times (productivity of blue-collar workers). Nevertheless, SE workers (white-collar workers) use other kinds of resources that are not only the hours of work indicated in the work contract [3]. An example of other "non-time" input used by them is knowledge and other intangible resources such as skills, competences… hence Drucker [77] named them: *knowledge workers*. Nevertheless, time is a resource that is used only once and if it is not used is equally consumed, hence the great importance of time for these workers. In addition, when working with time units it may be necessary to distinguish between nominal and hours actually worked on producing the outputs. Given the intellectual load of SE jobs, many workers work hours outside their schedule in order to complete assignments: for example an idea that solves a problem can appear at any time [70]. Thus, the inputs used in the obtained primary studies do not take into account these characteristics of workers in SE, and time is considered as the main factor. However, and in the face of this fact, research literature is developing strategies to include a broad set of methods in software development personnel issues (e.g. [2, 78]). As a result of this, future efforts will be devoted to integrate these new and interdisciplinary aspects into SE metrics.

On the other hand, outputs can be divided into two groups. The first group is focused on source code delivery. This trend can be useful in certain environments such as in software factories where the core business is coding the designs and requirements

demanded. However, this measure is not applicable to all SE tasks (it cannot be applied to software project managers for instance). In the other group, the use of completed tasks opens a parallel track for productivity measurement because it can be universally used – any job includes tasks to be performed, usually in their job description. Nevertheless, the use of this output leads to the difficulty of assessing tasks. This requires a very different approach depending on the tasks assessed. So, these outputs do not provide a universal solution and exploration of new alternatives should be considered. In any case, this factor is present in classic literature (e.g. [79]) and in more recent efforts (e.g. [80]) and deserves to be considered as a valid output.

Generally speaking, results show that there are few studies that address SE productivity measurement. Furthermore, these studies normally use SLOC/Time as the standard measure. These measures, included at higher levels, fail to measure productivity in a precise manner and leave unmeasured much of the work done at lower levels where the granularity is greater. Therefore, these measures lose precision and perhaps their usefulness [81] in such environments. Additionally, the use of time units as the only measure of input presents two readings: time (1) is the only resource that is actually spent when performing knowledge intense tasks and (2) it is the only tangible input, precise and universally available to be used as a measure of input for productivity measurement, while the other inputs are intangible and vague or non universally defined.

Finally, a standardized production time for each task and operation would be needed if it is assumed that the time consumed is the only input required. This goal is very difficult to obtain for almost all tasks of SE. This difficulty is not applicable when measuring the productivity of a worker in the manufacturing industry because each operation has a certain time to take place [82]. Moreover, the availability of this measure does not mean it has to be exclusive, i.e. it can be combined with other units of measurement to generate alternative measures of productivity. Thus, from the viewpoint of the authors of this paper, further research about productivity measurement at the job level in SE is needed. These future studies may also shed light into other knowledge-intense jobs, which currently represent the largest group of workers in developed countries [82].

7. Conclusions

As a first conclusion, a SLR protocol for finding information on measuring productivity at the job level in the SE area has been adapted from SLR standards. This adaptation can be used in any future SLR with similar research objectives. It can also be improved and modified to suit specific needs within this research area. Regarding implementation, the primary studies obtained are scarce, but their analysis provides information useful to obtain an overview of state of the art. However, it would be interesting to conduct a wider SLR to expand the target population, for example a SLR for for measures of productivity of knowledge workers or for the service sector at the specific job description level.

Secondly, the results indicate that the research in this area is based on the same measures of inputs and outputs, and therefore on the same measures of productivity that are used at higher measurement levels (project, department, organization...). From the inputs standpoint, time is the main factor. From outputs, SLOC and tasks are the major choice. Thus, productivity measures at the specific job description level in SE have two variants: one focused on measuring the productivity of a unit of product (SLOC/Time) and the other focused on planning project units (Tasks Completed/Time). The first variant, the primary goal of which is to measure the efficiency of a product's delivery, is frequently used at higher levels in SE. As an alternative to SLOC at these levels, other measures such as FP, which measure the software functionality that the product offers, are commonly used. The second variant, planning project units, is less frequently used at higher levels of productivity measurement.

Thirdly, the lack of factors related to quality in the measurement of productivity is alarming. With respect to this element, Drucker [82] indicated that quality is a factor that determines the productivity of a knowledge worker, therefore the productivity of these workers is a matter of quality and not just of quantity. In addition, this element is important in any job, but it is vitally important in knowledge jobs. It is also necessary to include in productivity measurement other dimensions such as customer satisfaction, responsibility of workers and importance of each task, the perceived productivity of workers and absenteeism [3]. In addition to these aspects, other specific SE elements could be considered; for example the reuse of source code and other products, the effect of the work environment, skills and job experience, organizational culture or characteristics of the project.

In conclusion, given the exploratory nature of this paper, there are numerous lines of research that could be carried out based on the results obtained. One of them is understanding software engineers' perception of productivity. With this perspective it would be possible to construct productivity measures taking into account firsthand experience and not drawing directly from academic research. Another line of research is the development of new measures of productivity using new criteria of inputs and outputs, as well as various combinations thereof. These measures could increase the accuracy in measuring productivity at the job level and shed light on new alternatives in other productive activities. Finally, it is necessary to increase the number of factors used in the measurement of productivity because any change in them can affect, positively or negatively, the productivity of software engineers.

References

[1] D. Dalcher, Supporting software development: enhancing productivity, management and control, *Software Process: Improvement and Practice* **11**(6) (2006) 557-559.

[2] R. Colomo-Palacios, E. Tovar-Caro, Á. García-Crespo and J. Gómez-Berbís, Identifying Technical Competences of IT Professionals: The Case of Software Engineers, *International Journal of Human Capital and Information Technology Professionals* **1**(1) (2010) 31-43.

[3] Y. W. Ramirez and D. A. Nembhard, Measuring knowledge worker productivity: A taxonomy, *Journal of Intellectual Capital* **5**(4) (2004) 602-628.

[4] J. S. Challa, A. Paul, Y. Dada, V. Nerella, P. R. Srivastava, and A. P. Singh, Integrated Software Quality Evaluation: A Fuzzy Multi-Criteria Approach, *Journal of Information Processing Systems* **7**(3) (2011) 473-518.

[5] SWEBOK: Software Engineering Body of Knowledge. 2004, IEEE Computer Society.

[6] A. MacCormack, C. F. Kemerer, M. Cusumano and B. Crandall, Trade-offs between Productivity and Quality in Selecting Software Development Practices, *IEEE Softw.* **20**(5) (2003) 78-85.

[7] S. Lee and R. C. Schmidt, Improving application development productivity in Hong Kong, in *Information technology and challenge for Hong Kong*, ed. J. M. Burn and B. M. Martinsons (Hong Kong University Press, 1997).

[8] M. Asmild, J. C. Paradi and A. Kulkarni, Using data envelopment analysis in software development productivity measurement, *Software Process: Improvement and Practice* **11**(6) (2006) 561-572.

[9] B. A. Kitchenham and E. Mendes, Software Productivity Measurement Using Multiple Size Measures, *IEEE Trans. Software Eng.* **30**(12) (2004) 1023-1035.

[10] P. C. Pendharkar, Scale economies and production function estimation for object-oriented software component and source code documentation size, *European Journal of Operational Research* **172**(3) (2006) 1040-1050.

[11] R. D. Banker, S. M. Datar, and C. Kemerer, Factors Affecting Software Maintenance Productivity: An Exploratory Study, in *Proc. Eighth International Conference on Information Systems (ICIS-1987)*, Pittsburgh, Pennsylvania, 1987 pp. 160–175.

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

[13] M. Tsunoda, A. Monden, H. Yadohisa, N. Kikuchi and K. Matsumoto, Software development productivity of Japanese enterprise applications, *Information Technology and Management* **10**(4) (2009) 193-205.

[14] R. D. Banker, S. M. Datar and C. F. Kemerer, A model to evaluate variable impacting the productivity of software maintenance projects, *Management Science* **37**(1) (1991) 1-18.

[15] F. Quesnay, Analyse de la formule arithmétique du tableau économique de la distribution des dépenses annuelles d'une nation agricole, *Journal de l'Agriculture, du Commerce & des Finances* (1766) 11-41.

[16] J. Jefferys, S. Hausberger and G. Lindblad, Productivity in the distributive trade in Europe: wholesale and retail aspects (Organisation for European Economic Co-operation 1954).

[17] A. Hernández-López, R. Colomo-Palacios, Á. García-Crespo and F. Cabezas-Isla, Software Engineering Productivity: Concepts, Issues and Challenges, *International Journal of Information Technology Project Management* **2**(1) (2011) 37-47.

[18] T. Dybå, and T. Dingsøyr, Empirical studies of agile software development: A systematic review, *Information and Software Technology* **50**(9-10) (2008) 833-859.

[19] P. Tomaszewski and L. Lundberg, Software development productivity on a new platform: an industrial case study, *Information and Software Technology* **47**(4) (2005) 257-269.

[20] S. Koch, Exploring the effects of SourceForge.net coordination and communication tools on the efficiency of open source projects using data envelopment analysis, *Empirical Software Engineering* **14**(4) (2009) 397-417.

[21] A. J. Albrecht and J. E. Gaffney, Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, *IEEE Trans. Software Eng.* **9**(6) (1983) 639-648.

[22] C. Melo, D. S. Cruzes, F. Kon and R. Conradi, Agile Team Perceptions of Productivity Factors, in *Proc. AGILE Conference (AGILE'2011)*, Salt Lake City, USA, 2011, pp. 57-66.

[23] Y. Wang, On the Cognitive Informatics Foundations of Software Engineering, in Third IEEE International Conference on Cognitive Informatics. 2004, IEEE Computer Society, pp. 22-31.

[24] P. R. Srivastava and M. P. Ray, Design and Sensitivity Analysis of a Formal Test Process Model, *International Journal of Recent Trends in Engineering* **1**(1) (2009) 67-72.

[25] C. B. Seaman, Qualitative Methods in Empirical Studies of Software Engineering, *IEEE Trans. Software Eng.* **25**(4) (1999) 557-572.

[26] J. J. Treinen and S. L. Miller-Frost, Following the sun: Case studies in global software development, *IBM Syst. J.* **45(**4) (2006) 773-783.

[27] D. Anselmo and H. Ledgard, Measuring productivity in the software industry, *Commun. ACM* **46**(11) (2003) 121-125.

[28] L. Nachum, Measurement of productivity of professional services, *International Journal of Operations and Production Management* **9**(9/10) (1999) 922-950.

[29] J. Rose, K. Pedersen, J. H. Hosbond and P. Kræmmergaard, Management competences, not tools and techniques: A grounded examination of software project management at WM-data, *Information and Software Technology* **49**(6) (2007) 605-624.

[30] A. Trendowicz and J. Münch, Factors Influencing Software Development Productivity - State of the Art and Industrial Experiences, in Advances in Computers, V. Z. Marvin, Editor. 2009, Elsevier. pp. 185-241.

[31] E. Chrysler, Some basic determinants of computer programming productivity, *Commun. ACM* **21**(6) (1978) 472-483.

[32] A. J. Albrecht, Measuring application development productivity. in *Proc. Joint SHARE/GUIDE/IBM Application Development Symposium*, Monterey, California, 1979, pp. 83-92.

[33] I. Vessey and R. Weber, Some factors affecting program repair maintenance: an empirical study, *Commun. ACM* **26**(2) (1983) 128-134.

[34] A. J. Thadhani, Factors affecting programmer productivity during application development, *IBM Syst. J.* **23**(1) (1984) 19-35.

[35] M. A. Cusumano and C. F. Kemerer, A quantitative analysis of U.S. and Japanese practice and performance in software development, *Management Science* **36**(11) (1990) 1384-1406.

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

[37] W. Scacchi, Understanding Software Productivity, in *Software Engineering and Knowledge Engineering: Trends for the Next Decade*, ed. W. D. Hurley (World Scientific Pub Co Inc, 1994). pp. 293-321.

[38] J. Gaffney, Software reuse-key to enhanced productivity: some quantitative models, *Information and Software Technology* **31**(5) (1989) 258-267.

[39] R. D. Banker and R. J. Kauffman, Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study, *MIS Quarterly* **15**(3) (1991) 375-401.

[40] S. Purao and V. Vaishnavi, Product metrics for object-oriented systems, *ACM Computing Surveys* **35**(2) (2003) 191-221.

[41] B. A. Kitchenham, What's up with software metrics? - A preliminary mapping study, *Journal of Systems and Software* **83**(1) (2010) 37-51.
[42] S. L. Pfleeger, Software Metrics: Progress after 25 Years?, *IEEE Softw.* **25**(6) (2008) 32-34.
[43] C. Bellini, R. Pereira and J. Becker, Measurement in software engineering: from the roadmap to the crossroads, *Int. J. Software Engineer. Knowledge Engineer.* **18**(1) (2008) 37-64.
[44] K. Petersen, Measuring and predicting software productivity: A systematic map and review, *Information and Software Technology* **53**(4) (2011) 317-343.
[45] B. W. Boehm and R. Ross, Theory-W Software Project Management Principles and Examples, *IEEE Trans. Software Eng.* **15**(7) (1989) 902-916.
[46] C. J. Dale and H. van der Zee, Software productivity metrics: who needs them?, *Information and Software Technology* **34**(11) (1992) 731-738.
[47] E. Gummesson, Quality dimensions: what to measure in service organizations, in *Advances in services marketing and management*, ed. T. A. Swartz, D. E. Bowen, and S. W. Brown, (JAI Press, 1992), pp. 64-78.
[48] S. Wagner and M. Ruhe. A Systematic Review of Productivity Factors in Software Development, in *Proc. 2nd International Workshop on Software Productivity Analysis and Cost Estimation (SPACE 2008)*, Beijing, China, 2008.
[49] B. W. Boehm, *Software Engineering Economics* (Prentice Hall PTR, Upper Saddle River, NJ 1981).
[50] C. E. Walston and C. P. Felix, A method of programming measurement and estimation, *IBM Syst. J.* **16**(1) (1977) 54-73.
[51] I. Sommerville, *Software Engineering* (Addison-Wesley, Reading, MA 2010).
[52] O. Gómez, H. Oktaba, M. Piattini and F. García, A Systematic Review Measurement in Software Engineering: State-of-the-Art in Measures, in *Software and Data Technologies*, ed. J. Filipe, B. Shishkov and M. Helfert, Springer Berlin Heidelberg, 2008, pp. 165-176.
[53] P. R. Srivastava, C. Mittal, A. Rungta, V. Malhotra and G. Raghurama, Non Homogenous Poisson Process Model for Optimal Software Testing Using Fault Tolerance, *MIS Review* **15**(2) (2010) 75-90.
[54] M. Jørgensen and M. Shepperd, A Systematic Review of Software Development Cost Estimation Studies, *IEEE Trans. Software Eng.* **33**(1) (2007) 33-53.
[55] I. A. Al-Darrab, Relationships between productivity, efficiency, utilization, and quality, *Work Study* **49**(3) (2000) 97-104.
[56] B. W. Boehm, Improving Software Productivity, *Computer* **20**(9) (1987) 43-57.
[57] D. S. Sink, T. C. Tuttle and S. J. DeVries, Productivity measurement and evaluation: what is available?, *National Productivity Review* **3**(3) (1984) 265-287.
[58] B. A. Kitchenham, Guidelines for performing Systematic Literature Reviews in Software Engineering. 2007, Keele University and University of Durham.
[59] E. Paiva, D. Barbosa, R. Lima, and A. Albuquerque, Factors that Influence the Productivity of Software Developers in a Developer View, *in Innovations in Computing Sciences and Software Engineering*, ed. T. Sobh and K. Elleithy, (Springer Netherlands, 2010), pp. 99-104.
[60] W. Afzal, R. Torkar and R. Feldt, A systematic review of search-based testing for non-functional system properties, *Information and Software Technology* **51**(6) (2009) 957-976.
[61] T. DeMarco and T. Lister. Programmer Performance and the Effects of the Workplace. in *Proc. 8th International Conference on Software Engineering*, Institute of Electrical and Electronics Engineers, New York, 1985, pp. 268-272.

[62] D. Bisant and J. Lyle, A two-person inspection method to improve programming productivity, *IEEE Trans. on Software Eng.* **15**(10) (1989) 1294-1304.

[63] K. Sherdil and N. H. Madhavji. Human-Oriented Improvement in the Software Process. in *Proc. Fifth European Workshop on Software Process Technology*, Berlin, 1996, pp. 145-166.

[64] R. B. Kieburtz et al., A software engineering experiment in software component generation, in *Proc. 18th International Conference on Software Engineering*, Berlin , Germany, 1996 pp. 542-552.

[65] X. Zhong, N. H. Madhavji and K. El Emam, Critical factors affecting personal software processes, *IEEE Soft.* **17**(6) (2000) 76-83.

[66] F. Padberg and M. M. Müller. Analyzing the cost and benefit of pair programming, in Proc. *Ninth International Software Metrics Symposium*, Sydney, Australia, 2003, pp. 166-177.

[67] T. DeMarco, Controlling Software Projects: Management, Measurement, and Estimates (Prentice Hall PTR, Upper Saddle River, NJ 1986).

[68] E. Koss and D. A. Lewis, Productivity or efficiency— Measuring what we really want, *National Productivity Review* **12**(2) (1993) 273-284.

[69] C. E. Craig and R. C. Harris, Total productivity measurement at the firm level, *Sloan Management Review* **14**(3) (1973) 13-29.

[70] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin,K. El Emam and J. Rosenberg, Preliminary guidelines for empirical research in software engineering, *IEEE Trans. Software Eng.* **28**(8) (2002) 721-734.

[71] F. P. Brooks Jr., *The Mythical Man-Month: Essays on Software Engineering* (Addison-Wesley Professional, Reading, MA 1995).

[72] H. Sackman, W. J. Erikson and E. E. Grant, Exploratory experimental studies comparing online and offline programming performance, *Commun. ACM* **11**(1) (1968) 3-11.

[73] B. W. Boehm, An Experiment in Small-Scale Application Software Engineering, *IEEE Trans. Software Eng.* **7**(5) (1981) 482-493.

[74] G. J. Myers, Reliable software through composite design (Petrocelli/Charter 1975).

[75] P. Drucker, *The Landmarks of Tomorrow* (Harper & Row, New York 1959).

[76] R. Colomo-Palacios, C. Casado-Lumbreras, P. Soto-Acosta and Á García-Crespo, Using the affect grid to measure emotions in software requirements engineering, *Journal of Universal Computer Science* **17**(9) (2011) 1281-1298.

[77] T. K. Abdel-Hamid, The Dynamics of Software Project Staffing: A System Dynamics Based Simulation Approach, *IEEE Trans. Software Eng.* **15**(2) (1989) 109-119.

[78] H. S. Bok and K. S. Raman, Software engineering productivity measurement using function points: a case study, *Journal of Information Technology* **15**(1) (2000) 79-90.

[79] N. E. Fenton and M. Neil, Software metrics: success, failures and new directions, *Journal of Systems and Software* **47**(2-3) (1999) 149-157.

[80] P. Drucker, Knowledge-Worker Productivity: The Biggest Challenge, *California management review* **41**(2) (1999) 79-85.