Software Productivity Measurement and Prediction Methods: what can we tell about them?

Wladmir Araujo Chapetta[1, 2], Guilherme Horta Travassos[1]

[1]PESC/COPPE – Federal University of Rio de Janeiro (UFRJ) Cidade Universitária
Rio de Janeiro – RJ – Brazil

[2]DMTIC/DIMCI/INMETRO – National Institute of Metrology, Quality and Technology
Rio de Janeiro – RJ – Brazil

wachapetta@inmetro.gov.br, ght@cos.ufrj.br

Abstract. *An adequate way of making software organizations to remain competitive is to ensure their innovative capacity and the continuous increasing of their software process productivity with quality. Indeed, the ability on increasing the software productivity relies on among other issues in the organization's measurement and prediction capacity. Productivity refers to the rate at which a company produces goods, and its observation takes into account the number of people and the amount of other necessary resources to produce such goods. However, it is not clear how productivity can be observed when the product is software. Therefore, this work presents the results of an investigation regarding software productivity measurement and prediction methods. A previous systematic literature review was evolved and re-executed, limited to the year 2013. It allowed the identification of 89 new primary studies evidencing that: (1) ratio-based and weighted factors analyses still represent most of the methods applied to measure, describe and interpret software productivity; (2) 24 factors present evidence of influencing productivity, and; (3) SLOC-based measures, despite the criticism and issues associated with these sort of measurements, are the most common measures used in the studies.*

1. Introduction

Innovative capacity is the ability of an organization to produce and monetize a flow of improvements that would be reached by modifying attributes of its processes, products or services. These improvements are concerned with the value changing of the goods and services according to the customers' needs and their quality requirements. However, the performance changing must be perceived and, therefore, its effects must be measured and controlled.

Organizations pursue performance management as a good practice to produce the expected outcomes under repetitive conditions, preferably consuming resources without slacks or losses. Quality management system methodologies, i.e., CMMI (Monteiro and de Oliveira, 2011), MPS.BR (Santos *et al.*, 2010) and ISO9000 (Hwang and Kim, 2005), consider effectiveness and efficiency as proxies to identify and achieve the ongoing organizational performance.

Furthermore, the ISO 9000 family of standards argues that the performance measurement establishes methods to measure the effectiveness and efficiency of processes. The term effectiveness is concerned with the extent to which the planned activities are performed and the expected results achieved, whereas efficiency is concerned with the relationship between the achieved results and used resources. Therefore, in this scenario, productivity is stated as a dimension of software process efficiency.

Productivity usually refers to the ratio of the output to the input (de Aquino Junior and Meira, 2009a; Monteiro and de Oliveira, 2011; Petersen, 2011). In software processes, it has been studied at least for two decades, and it is still important and recognized research theme (de Aquino Junior and Meira, 2009a; Monteiro and de Oliveira, 2011; Petersen, 2011). In contrast to manufacturing, software productivity measurement and prediction are complex, and the methods usually proposed in the technical literature do not attract practitioners to their use. In fact, software productivity and prediction still present some issues in current research including that some of its factors and relationships are unknown. The technical literature confirms the scientific interest on this topic; although many problems represent open questions and many myths persist in confounding the SE community (de Aquino Junior and Meira, 2009a).

This paper updates and extends the findings of a previous systematic literature review on this topic aiming at to contribute to the evolution of software productivity measurement and prediction knowledge. It identified 89 additional primary studies, more information about the used measures besides selecting a set of factors affecting software productivity. We believe it represents a useful piece of knowledge that can support practitioners and researchers on deciding about productivity regarding their software processes and future studies.

The remainder of this work presents results of the evolved systematic review protocol about software productivity measurement and prediction methods, as well as the empirically evidenced factors influencing productivity in software processes. After that, the current and previous findings are compared, aiming at to support an additional discussion on the issues that can affect future researchers in the topic. Section 2 presents a brief discussion on the theoretical background in software productivity, justifying our research. In section 3, the research protocol is detailed, as well as the searching and extraction procedures are presented. Section 4 presents the findings, describing the relevant information to answer the research questions. Section 5 discusses some limitations observed in this secondary study. At last, based on the results, conclusions are drawn and open issues presented to the SE Community.

2. Background

To measure software productivity is essential to improve and control the software development performance (Petersen, 2011). It plays a major role in process improvement initiatives because it allows the definition of a baseline for improvements, which can be re-evaluated and evolved once such improvements have been implemented. Furthermore, software productivity measurements are important benchmarks to determine the need for new improvements aiming at to keep the organization competitive. Within an organization, benchmarking is a relevant tool to identify the best performers and then learn from them (Petersen, 2011).

Nevertheless, software productivity prediction is concerned with quantifying (by estimating with minimum error) the expected level of productivity in the future. So, the software productivity prediction refers to determine whether corrective actions are needed and to discover which alternative improvement action would yield the best development performance results (Petersen, 2011).

The following works represent the theoretical background in software productivity supporting our initial discussions:

1. Putnam and Myers (1991) notice software staffing profiles follow the well-known Rayleigh distribution and propose the metric called process productivity. The proposed metric is based on years of experience in R&D project management at the U.S. armed forces, and later in General Electric Co. involving projects developed in the United States, England, Australia and Japan.
2. Scacchi (1991) reviews the technical literature regarding software productivity from the perspective of confounding factors and threats to validity of primary studies, such as: (1) Poor definition of measures (e.g. measuring lines of code versus the effort related to all development life-cycle activities); (2) Unclear root-cause effect for changes in software productivity measures for function points, and; (3) Ignorance of a combined effect of software productivity factors (Petersen, 2011).
3. Dale and van der Zee (1992) discuss software productivity measures from different perspectives (i.e. development, user, and management). Concerning lines of code, the authors point out that LOC is only one part of the valuable outputs produced during software development. The same applies to the effort related to producing such lines of code.
4. de Aquino Junior and Meira (2009) review and categorize productivity metrics and measurement studies published in the technical literature. The authors point out that the most common used measure is the SLOC (Source Lines of Code) discussing the critics and paradox about it.
5. Hernandez-Lopez *et al.* (2011) describe an overview of the state of the art in productivity measurement in software engineering. The authors undertake a systematic literature review (SLR) to assess the current inputs and outputs presented in the technical literature to discuss productivity measurement with software practitioners.
6. Petersen (2011) identifies 38 primary studies regarding software productivity measurement and prediction methods through a systematic literature review. Based on them, the author proposes a classification scheme for empirical research in software productivity.
7. Hernandez-Lopez *et al.* (2013) present software engineering productivity general concepts, issues and the challenges needing SE researchers' attention. The authors point out the need to reach a consensus on influencing software

productivity factors and recognize useful sets of inputs and outputs for using in the software productivity measurement.

These studies show their results focusing on different purposes and have been performed in various moments. Looking at them together to infer which results could be combined to produce reliable nowadays scenarios is not straightforward. Thus, the motivation for our study is to provide an updated and unified viewpoint, considering the questions related to measures, methods, factors and empirical propositions on software productivity measurement and prediction.

The next sections aim to describe the research design, confirm and update the results obtained in Petersen (2011), replicating and evolving the original systematic literature review protocol, as well as presenting updated results and detaching new findings on software productivity measurement and prediction.

3. Research Method

3.1. Systematic Review Protocol

Our secondary study protocol is based on Biolchini *et al.* (2005). To organize the search string, it uses the PICO (Population, Intervention, Comparison, and Outcome) approach (Pai *et al.*, 2004). In our case, complying the PICO approach, there is no comparison available. The main difference between our protocol and Petersen (2011) is the population enlargement. Thus this study's goal is mainly to characterize software productivity measurement and prediction.

Due to the interchangeable quoting of productivity, efficiency, performance and, often, effectiveness (de Aquino Junior and Meira, 2009a; Monteiro and de Oliveira, 2011; Petersen, 2011) in the technical literature, our study adopts the following definitions for these concepts:

- Productivity is the ratio describing the relationship between the outcomes and the resources used to produce them.
- Efficiency is a relative concept. It compares what has been produced, considering the available resources, and what could be produced with the same resources. Thus, when comparing two productivity measures, we can observe the most efficient entity.
- Effectiveness is the ability to achieve something or producing the intended result.

Thus, productivity and efficiency are interchangeable terms without loss of meaning, but it is not necessarily true when effectiveness is compared with the two other terms. The effectiveness is not being taken into account in our study to preserve the generalization and analysis criteria of Petersen (2011).

3.1.1. Research Questions

The research questions focus on identifying measurement and prediction methods used to measure the efficiency or productivity or performance of software processes. In complement, the questions intend to verify the existence of measures and factors that can promote/be used in the measurement of efficiency or productivity in software processes. In our work, the research questions were kept to allow additive aggregation with Petersen (2011). Table 1 shows the research questions (RQ's).

Table 1. Research Questions

Main Question:	RQ1: Is there any evidence on the accuracy/usefulness of the software productivity or efficiency measurement and prediction methods?	
	Secondary Questions:	RQ1.1: Is there any measure that can support the measurement and prediction methods? If so, is it always true for different software processes?[1]
		RQ1.2: Which factors (or drivers or constructs) have been used to define/compose/describe such measurement and prediction methods?[1]
Alternative Questions:	RQ2: What recommendations can be given methodologically to improve software productivity (or efficiency) studies and the packaging and presentation of such studies?	
	RQ3: Are there any general propositions, hypotheses or theories that have been used to justify software productivity (or efficiency) studies performed by Software Engineering researchers? If so, which ones can be used to define a research agenda on software productivity or efficiency studies for the Software Engineering community? [1]	
Mapping Questions:	RQ4: How has the frequency of methods related to measuring and predict software productivity changed over time?	
	RQ5: How is the frequency of published research distributed within the structure of software productivity research?	

[1] additional research questions

3.1.2. Search Strategy

Essentially, Biolchini *et al.* (2005) define that a search strategy is composed of search strings and engines. Table 2 shows the set of engines. Our study adds the Scopus search engine to the original set of engines. Petersen (2011) reported that the previous versions of the search engines were not able to handle complex search strings. In our study, this procedure was not necessary due the evolution of the search engines and the facilities presented in the current versions. However, his search strings could not be reused, because the grammars of the common search engines have changed since the execution of the original searches. It was reformulated and adapted by combining split terms and logical operators. Thus, the following search string was used in our study:

***Title or Abstract**: (("software process" OR "software development" OR "software engineering processes" OR "business information system" OR "software maintenance" OR "software project" OR "open source project" OR "OSS project") AND (productivity OR efficiency OR "process performance" OR "development performance" OR "software performance" OR "project performance" OR "prediction method" OR "measurement method") AND (measure OR metric OR model OR predict OR estimate OR measurement OR estimation OR prediction) AND (empirical OR validation OR evaluation OR experiment OR example OR simulation OR analysis OR study OR interview))*

All included papers in the original protocol were read and used as control ones, i.e., they should be found in our subsequent SLR trials and included. This arrangement was adopted to guarantee search quality, considering the coverage and scope of the known software productivity literature to minimize the researcher bias.

3.1.3. SLR Protocol's Criteria

According to Biolchini *et al.* (2005), the SLR Protocol's Criteria are described to define how the researchers will judge the articles returned by the search engines and therefore decide on their pertinence and inclusion. The selection criteria are divided into four categories: selection, inclusion, exclusion, and acceptance.

The Selection Criteria allow making explicit under which conditions an article can be selected. In our study, the scientific publications were looked for in peer-reviewed journals and conferences available on the web through search engines.

Table 2. Search Engines

Name	URL
Web of Science	http://www.webofknowledge.com
IeeeXplore	http://ieeexplore.ieee.org/
EngineeringVillage	http://www.engineeringvillage.com
ACM Digital Library	http://dl.acm.org/
Scopus	http://www.scopus.com/

The exclusion criteria determine the parameters for excluding articles out of the research protocol scope. They are: (1) Software productivity (efficiency or performance) measurement or prediction is not the main focus, OR; (2) Talking about direct measures that are not combined with other measures, and thus no claims regarding software productivity, efficiency and performance are possible, OR; (3) It is a secondary study, that is excluded from the current synthesis and results, OR; (4) Talking about measures for single techniques, which do not give an indication on how well the organizations or teams perform the measurement (whether in the overall process level or within a development phase, e.g., a primary study measuring defect rate without comparing inspection techniques or tools), OR; (5) The article does not present any evaluation (proof of concept, case study or experimental study).

The inclusion criteria determine the parameters for including studies in the next review steps. In this review, studies must be included whether talking about: (1) Software performance, productivity or efficiency of software development, software processes, software projects, software developers, OR; (2) Relations among different direct measurements or attributes (like size and effort, or reliability and effort, lead-time and resources, or multivariate relations) that are used to characterize the performance, productivity or efficiency of software development, software processes, software projects or software developers, AND; (3) Proof of concept, experiment, case study, example or other empirical research methods showing its applicability and usefulness/accuracy. It is also reasonable to consider papers based on real-world data.

The acceptance criteria are used to determine how the reviewers have to proceed in selecting the articles for full reading. In this review, acceptance criterion is: Two distinct reviewers evaluate each study, and whether both reviewers agreed on excluding the study, it is excluded. Otherwise, it is included.

3.2. Study Selection Procedure

The study selection procedure can be divided into 3 phases according to Biolchini *et al.* (2005) and Dybå *et al.* (2007): (1) Performing the search in databases; (2) Excluding articles based on the title and abstract; and, (3) Including articles based on the full reading.

3.2.1. Phase 1: Performing the Searches

At this point of the review process, all criteria have already been defined, and behavioral decisions have already made under the risk of re-evaluating and re-calibrating the research protocol. Otherwise, trade-offs or adjustments would be necessary. Members of the Experimental Software Engineering Group at COPPE/UFRJ have adopted the following criterion to perform the search in databases: The search will always be limited to papers before the current year of performing. This decision aims to minimize the changes of returned papers while performing different SLR trials.

For comparison purposes, the search string of Petersen (2011) was repeated aiming to find not previously identified articles. The original search was re-executed in April 14th, 2014 and limited to papers published until December 2013.

After repeating the original search, the new search string was piloted, evolved, and finally performed on August 26th, 2014 and was also limited to papers published until December 2013. The total of returned articles was 7087.

Due the availability of the original research protocol (Petersen, 2011) and all the materials (Dr. Kai Petersen kindly shared all this information with us), we decided not to review all studies previously returned, removed or included by Petersen (2011) in our study. In such cases (previously evaluated articles), Petersen (2011)'s actions just were identified and reproduced, i.e., there was no judgment. All new article entries were treated according to our protocol, even being material published before 2009. Indeed, the effort to identify those studies was lower than to execute the extraction procedure again.

After merging duplicates and remove previously evaluated articles, the total of articles for the next phase was 3209.

3.2.2. Phase 2: Excluding Articles Based on Title and Abstract

This phase consists of assessing the articles on their titles and abstracts. In this study, the number of selected articles for full reading was 318. At this point, the exclusion criteria are predominantly applied to the articles.

3.2.3. Phase 3: Including Articles based on Full Reading

This phase consists of evaluating primary studies on their full reading. The number of selected articles for data extraction became 89 studies. At this point, the inclusion criteria are predominantly applied to the articles.

In Phases 2 and 3, the articles not accessible and which the authors could not respond or be located were removed and properly reported in the study package.

3.2.4. Data Extraction

The data extraction form respects the content and structure of Petersen (2011) to guarantee that new findings can be properly aggregated. However, some minor adjustments were necessary to accommodate the new information about our additional research questions.

Each selected article was classified regarding its approach, purpose, abstraction, type of evaluation, evaluation criteria, research method and empirical rigor, based on the scheme and quality rating criteria presented in Petersen (2011).

4. Review and Findings

4.1. Methods in Software Productivity Research (RQ1)

The list of newly selected articles and all information extracted is presented in Chapetta (2016a) that summarizes the measurement methods, evaluation and quality rating of the selected articles. More information about the methods and classification scheme can be found in Petersen (2011). Seven new ways to measuring and predict software productivity have been identified: simulated annealing, agent-based model, neural networks, QEST nD model and Cumulative sum test. Such methods are described as follow:

- Value-based Measurement assumes the performance metric must attempt to measure the changes in the value of relevant stakeholders, and assumes that if an organization is successful in capturing value for stakeholders, then it can perceive and react to conflicts, suggesting that value can be more efficiently delivered (Boehm, 2003). In this review, only de Aquino Junior and Meira (2009b) employed value-based measurement, presenting a proof of concept and proposing a methodology to implement and execute a measurement process that theoretically would support practitioners to define a measure of productivity based on the value perceived by pre-defined stakeholders.
- Machine Learning Techniques cover a high number of methods. However, Bibi *et al.* (2008) presents a combination of two of them (Association Rules and Classification and Regression Trees) to predict software productivity increasing/decreasing. In this study, the methods are combined to deliver a productivity estimation framework. The proposed conjunction of methods has the ability of learning and modeling associations to describe unknown relationships and combining probabilistic methods to find out a model. Thus, the combined model can reveal the way in which particular attributes and facts can increase or decrease software productivity. It motivated us to characterize the study through a dual classification.
- Simulated Annealing is an algorithm based on the analogy between the annealing of solids simulation and the problem of solving large combinatorial optimization problems. It works by emulating the physical process whereby a solid is slowly cooled so that when eventually its structure is frozen, this

happens at a minimum energy configuration. So, Celik *et al.* (2011) employ this method to propose a solution for human resources allocation in software projects at the individual level, providing a guideline supporting various factors to predict the productivity of software developers based on COCOMO II.

- Neural Network: It is a technique of computing and signal processing inspired by a network of biological neurons. Therefore, Lopez-Martin *et al.* (2013) modeled and trained a neural network with a data set of 140 individual software projects developed between the years of 2005 and 2008 with practices based on the Personal Software Process (PSP). Next, it compared the accuracy obtained by the neural network to those by fuzzy logic and statistical regression models.
- Agent-based Models: Kang *et al.* (2011) define a framework based on a simulation method to propose a solution for workforce assignment, applying an agent-based model to select the best possible workforce allocations and estimate the productivity as well as the long-term organizational performance.
- QEST nD model: This model describes the overall process performance as a combination of any considered dimension, calculated as the weighted sum of the applied metrics. According to Ardagna *et al.* (2010), it is decoupled from specific development processes, allowing multi-process, multi-project performance analysis.
- Cumulative sum test: consists of a sequential test of changes accumulating evidence as each new sample is taken and has been used for the detection of changes in stochastic variables, such as mean value and variance of a Gaussian process. Ramil and Lehman (2001) perform such test permitting the examination of whether productivity in its project had changed over an 11 years period, by using empirical data of software modules.

By adding the newly selected articles to the Petersen (2011)'s original set, the total of articles is 127.

4.1.1. Measures Used in the Methods (RQ1.1)

When looking at the used measures in these 127 studies, it reveals that the new studies are using SLOC-based (29 studies) an FP-based (24 studies) productivity measures. The most common input measure continues to be effort (man-hour or derivations), with new 31 citations (34 in total). As the citations of each measure represent less than 27% of 127 studies, it seems that issues related to software processes output and input did not reach the consensus of researchers and practitioners yet. Thus, the general discussion and analysis presented in Petersen (2011) are still valid, even considering the number of selected studies is considerably greater.

4.1.2. Factors Observed in the Methods (RQ1.2)

Petersen (2011) reports being difficult to capture factors affecting software productivity due to the small size of the studies. However, we revisited all the 127 selected studies looking for factors empirically evaluated and presenting specific confidence levels or significant difference among the treatments relating such factors to software productivity. Based on our investigation, we observed 83 different factors that were empirically evaluated and demonstrated to affect software productivity. For the sake of confidence, only the factors presenting at least three studies supporting their effects have been considered for further investigation. The list of factors and their frequencies in the set of 127 selected studies are shown in Table 3.

Several of the listed factors are well-known and self-explanatory. Therefore, we are not discussing their impact and meaning in the scope of this paper. In this manner, "LOC-based" and "Functional" Size are listed separately. At this moment, we are considering both as capturing different aspects of software, even these aspects having a common denomination, in this case, the idea of "size". More details about evidence related to the factors can be seen in Chapetta (2016b).

Table 3. Factors affecting software productivity

Factor	Frequency	Factor	Frequency
Team Size	9	Software Risk Exposure and Management	4
SPI Adoption	7	Quality Methods Usage	4
Flexible Design and Reuse	7	Functional (FP-based) Size	4
LOC-based Software Size	7	Business Area	4
Personnel Capability	7	Team Dispersion	4
Active Stakeholder Participation	6	Communication Availability	3
Experience in Technology	6	Type of Language	3
Asset Complexity	5	Team Diversity	3
Team Cohesion	5	Domain Knowledge	3
Development Methodology	5	Expertise in Technology	3
Staff Availability and Allocation	4	Team Autonomy	3
Use of Tools	4	Precise Documentation Availability	3

4.2. Recommendations (RQ2)

Our results do not permit to infer about the intensity and significance of factors affecting the software productivity, yet, and we are going to deal with this concern in the nearest future. Nevertheless, if researchers do not understand a phenomenon, they should observe the available empirical evidence to get a better understanding of the research theme. Moreover, researchers have to ask themselves what methodological improvements must be implemented in their primary and secondary studies and what direction their investigations should take. In this research, these include discussing: (1) software process input and output measures and their relevance for stakeholders in different software processes; (2) aggregating available evidence, and; (3) how to quantify the necessary evidence to justify the investigation of a factor.

No evidence about the experimental packaging of primary studies has been found, not allowing to replicate any of the primary studies easily. Thus, an adequate packaging is of interest since the lack of such packages represents a significant issue on replicating primary studies, deserving attention from the methodological viewpoint.

At last, the data extraction in this review was difficult due to how the selected studies were reported scattering the relevant information in different sections (Petersen, 2011). Therefore, the adoption of some standards and the pre-classification of primary studies during reporting, e.g., by using a scheme as the proposed by Petersen (2011), could make more efficient to extract data in secondary studies.

4.3. About General Propositions (RQ3)

Even that we have found a considerable amount of evidence and factors, no general propositions, hypothesis or theories could be extracted from these studies. It indicates the absence of consensus about which and how the phenomena must be observed and better understood, i.e., there is no definition of essential questions to guide SE researchers in specific studies about software productivity measurement and prediction.

4.4. Methods Distributed Over Time (RQ4)

Like in Petersen (2011), our review divides its findings into the periods before 2001, and; between 2001 and 2008, and also adds a new period, between 2009 and 2013, aiming to enable the comparison of the achieved results of Petersen (2011) and new studies found. The set of new studies is composed of 18 studies published until 2001, 21 studies between 2001 and 2008. By aggregating these studies to results of Petersen (2011), there are new 77 selected studies in the period until 2008. Consequently, there are new 50 studies in the interval between 2009 and 2013. These numbers indicate an increasing on the empirical investigation of software productivity measurement and prediction methods in the last five years of our review.

Furthermore, our review extends the findings of Petersen (2011) presenting the following results: (1) until 2001, it includes 18 studies, three occurrences about weighting factors, eight about ratio-based measurement and two about simulation. (2) Between 2001 and 2008, there are nine references to ratio-based analysis, five about weighting factors and one about simulation. (3) From 2009 to 2013, there are 19 studies on weighting factors and eight about simulation.

4.5. Published Research (RQ5)

The weighting factor productivity and ratio-based measurements continue to be the most studied methods, each one presenting 33 studies. There is significant interest (20 studies) on using Data Envelopment Analysis (DEA) as a measurement method, but these studies do not allow comparison because there are different perspectives and first measures used among them. The period between 2001 and 2008 concentrates most of the studies (eight) regarding DEA.

The empirical rigor classification takes into account eight quality rating criteria, and they just check whether the description of certain aspects related to the empirical rigor of selected studies has been reported: Context, Data Collection, Validity, Construction, Control Group, Analysis, Findings, and Variables. Thus, the empirical rigor of a study is obtained by counting the presence of such aspects. Petersen (2011) suggests that these quality criteria and respective values reduce the reviewer bias due to their simplicity. Regarding the research method classification, it just divides in Evaluation (longitudinal/observational studies) and Validation (lab studies/experiments).

Although the time lapse and some new findings, our results reinforce the previous findings of Petersen (2011) on which the weighting productivity factors, simple input/output ratio measurement, and event-based models are the methods the SE community exploits. Other methods, such as Statistical Process Control, Bayesian Network, and others have been scarcely studied.

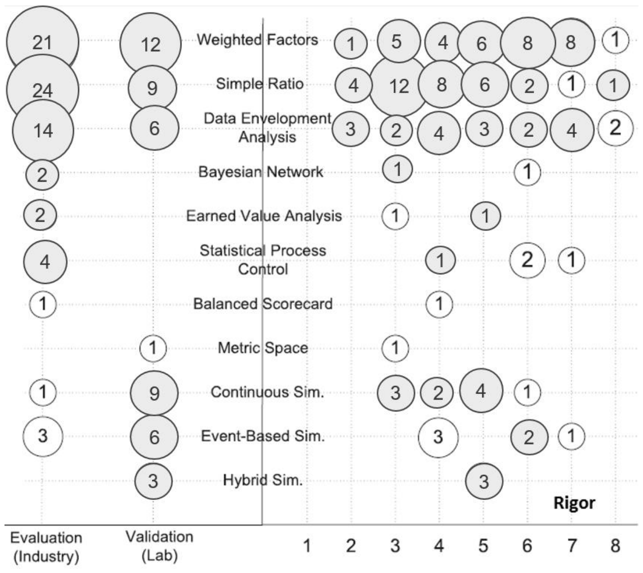

Figure 1. Evolved from Petersen (2011)

Figure 1 presents the distribution of methods found in the technical literature, using the same Petersen (2011)'s empirical rigor and research method classifications. Figures 1 and 2 adopt the same representation found in Petersen (2011), highlighting the information about the additional 89 selected studies in gray and showing how the information about the selected articles are distributed in the technical literature. Figure 2 detaches the new methods found.

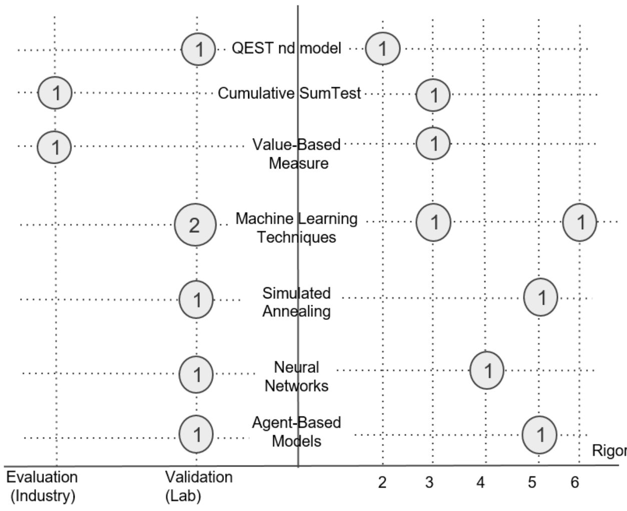

Figure 2. New methods found

5. Limitations

The main threat to validity of our study is the researcher bias, which has been minimized through peer-reviews where the researchers evaluated the resource questions, inclusion/exclusion criteria, data extraction and analysis of articles. The results of Petersen (2011) were used as a baseline to avoid that proposed changes in the research protocol could distort previous results. The risk of excluding relevant articles was reduced by being inclusive, i.e. the researchers included articles for detailed review when in doubt of their pertinence. Even studies with lower quality rating were included as in Petersen (2011).

A significant amount of returned articles led us to limit our search to 2013 and not to discuss other evaluation methods (survey, action research, etc.) beyond the original ones. These limits are two threats to validity. Nevertheless, by using the current versions of search engines, our study demonstrates that replication of secondary studies is feasible, and we can update these findings by performing our search restricted to the periods after 2013. Furthermore, the current results do not become less actionable in the short term, because this temporal gap does not make them invalid for other researchers. For the sake of time and effort, updates are going to be addressed in the nearest future.

From another perspective, this work used a simple criterion to select software productivity factors, which is based on our empirical perception of the factor strength relates to its minimum evaluation through three studies. This number aims to eliminate some factors that have been coincidentally investigated in only two different studies. It is a start point, but is it valid? What can give us at this time confidence on the influence of a software factor in productivity?

Furthermore, Petersen (2011) suggests that its quality criteria reduced the reviewer bias due to their simplicity. In fact, they are very simple and have easily allowed an adequate way for guiding the data extraction from primary studies, but the discriminatory power of evidence is noticeably jeopardized.

6. Conclusions

This work executed a bottom-up approach to identify software productivity measurement and prediction methods and influential factors through undertaking a Systematic Literature Review. It replicated and evolved review contributed to revealing a large number of studies not previously identified in Petersen (2011), although the discussions on issues and criticism regarding the software productivity measurement and prediction methods are slowly evolving since 2009. Apparently, researchers have invested on investigating weighting factors and ratio-based measurements by convenience (without defining a common research agenda) and, fitting the software project data to test their models and find out correlation among the factors under observation. It avoids evolving knowledge regarding predictors for productivity and to strength evidence by lacking aggregation possibilities, what highlight the need for continuously performing further (repetitive) studies.

Concerning the time lapse between Petersen (2011) and this work, the additional number of selected articles can justify the effort spent and the proposed modifications to the original research protocol Petersen (2011), so that our results enlarge and evolve the original results regarding sample frame. The current research protocol was able to select studies in the period before 2008, due to, probably, indexing and updating of content in a set of chosen databases. These studies strength some known phenomena and may point out which others need more rigor when being observed.

No agreement about directions in software productivity theme has yet been reached, and further investigations are still necessary. Currently, we are trying to express the expected effect and relationships among the factors and software productivity. This ongoing step aims to draw an initial road map, through the definition of a common theoretical model, researchers and practitioners may consider that in their studies and improvement initiatives in the industry. We hope this initial model may contribute to discuss a joint research agenda on software productivity measurement and prediction within the SE community.

Nevertheless, without a precise definition of a research agenda, how to achieve general models and theories that can be employed in the industrial setting? How to talk about improvement initiatives when there is no consensus among researchers and practitioners on what should be measured regarding software productivity? The answer to these questions can support the discussion of directions in the field of software and systems productivity research.

For the practitioners, the list of presented factors is based on empirically evaluated studies and represents a start point to observe software productivity in software organizations. It can be taken into account when managers and project leaders are planning their improvement initiatives and intend to measure productivity in their software projects.

Controlling and improving software productivity is admittedly decisive for successful software projects and competitive organizations (de Aquino Junior and Meira, 2009a; Monteiro and de Oliveira, 2011; Petersen, 2011). However, some primary issues (conceptual consensus, common research agenda) need to be discussed by the SE community to allow the increasing of the knowledge evolution pace regarding software productivity measurement and prediction in the face of contemporary software projects.

7. Acknowledgments

Many thanks to Prof. Kai Petersen, Blekinge Institute of Technology, Sweden for promptly sharing his SLR lab package with us. Prof. Travassos is a CNPq Researcher (grant 305929/2014-3).

References

Ardagna C, Damiani E, Frati F, Oltolina S, Regoli M and Ruffatti G, 2010, Spago4Q and the QEST nD Model: An Open Source Solution for Software Performance Measurement, IFIP AICT, Vol. 319, pp. 1-14.

de Aquino Junior, G S de; Meira, S, 2009a, Software Productivity Measurement: Past Analysis And Future Trends, In: Proc. 3th ICSETP.

de Aquino Junior GS and de Lemos Meira SR, 2009b, An Approach to Measure Value-Based Productivity in Software Projects, In: Proc of 9th QSIC., pp. 383-389.

Boehm B, 2003, Value-Based Software Engineering: Reinventing, SIGSOFT Softw. Eng. Notes, 28(2):3.

Bibi S, Stamelos I and Angelis L, 2008, Combining Probabilistic Models for Explanatory Productivity Estimation, Elsevier IST, pp. 656-669.

Biolchini, J, Mian, P G, Natali, A C, Travassos, G H, 2005, Systematic Review in Software Engineering: Relevance and Utility. Technical Report. PESC - COPPE/UFRJ. Brazil. http://www.cos.ufrj.br/uploadfiles/es67905.pdf

Celik, N.; Lee, S.; Mazhari, E.; Son, Y.-J.; Lemaire, R. and Provan, K. G. (2011), Simulation-based workforce assignment in a multi-organizational social network for alliance-based software development, SMPT 19(10), 2169 - 2188.

Chapetta,WA, 2016a, Supplementary Material of SLR-2014, https://goo.gl/1eGNme , Accessed in Jun 30, 2016.

Chapetta,WA, 2016b, Evidence (selected articles) vs Factors, https://goo.gl/zKC6PP , Accessed in Jun 30, 2016.

Dale, H. van der Zee, 1992, Software Productivity Metrics: Who Needs Them?, Elsevier IST, pp. 731–738.

Dybå T, Dingsøyr T, Hanssen GK, 2007, Applying Systematic Reviews to Diverse Study Types: An Experience Report, In: Proc. of the 1st ESEM 2007, pp. 225–234.

Hernandez-López, A.; Colomo-Palacios, R.; Garca-Crespo, A., Cabezas-Isla, F., 2011, Software Engineering Productivity: Concepts, Issues, and Challenges, IGI Global, Vol. 2, pp. 37-47

Hernandez-Lopez, A.; Colomo-Palacios, R.; Garcia-Crespo, A., 2013, Software Engineering Job Productivity - A Systematic Review, IJSEKE, Vol. 23, pp. 387-406.

Kang D, Jung J and Bae D-H, 2011, Constraint-based human resource allocation in software projects, Software-Practice and Experience, Vol. 41(5), pp. 551-577.

Hwang S-M and Kim H-M, 2005, A Study on Metrics for Supporting The Software Process Improvement Based on SPICE, LNCS,Vol. 3647, pp. 71-80.

Lopez-Martin, C.; Chavoya, A. and Meda-CAMPANA, M. E. (2013), Software development productivity prediction of individual projects applying a neural network, In: Proc. of 6th IMETI, pp. 47-52.

Monteiro L and de Oliveira K, 2011, Defining a Catalog of Indicators to Support Process Performance Analysis, Wiley JSME. Vol. 23(6), pp. 395-422.

Pai, M., McCulloch, M., Gorman, J. D., Pai, N., Enanoria, W., Kennedy, G. & Colford Jr, J. M. , 2004, Systematic Reviews and meta-analyses: An illustrated, step-by-step guide, The National Medical Journal of India, vol. 17, n.2.

Petersen K, 2011, Measuring and Predicting Software Productivity: A Systematic Map And Review, Elsevier IST, Vol. 53, pp. 317-343.

Putnam LH and Myers W, 1991, Measures for Excellence: Reliable Software on Time, within Budget. Prentice Hall Professional Technical Reference.

Ramil J and Lehman M (2001), Defining and applying metrics in the context of continuing software evolution, In: Proc. of 7th METRICS 2001, pp. 199-209.

Scacchi W, 1991, Understanding Software Productivity: A Knowledge-Based Approach, World Scientific IJSEKE, pp. 293–321.

Santos G, Kalinowski M, Rocha A, Travassos G, Weber K and Antonioni J, 2010, MPS.BR: A Tale of Software Process Improvement and Performance Results in the Brazilian Software Industry, In: Proc. of 7th QUATIC, 2010, pp. 412-417.