The Quest for Productivity in Software Engineering: A Practitioners Systematic Literature Review

Carlos Henrique C. Duarte
BNDES, Av. Chile 100, Rio de Janeiro, RJ, 20001-970, Brazil
cduarte@bndes.gov.br & carlos.duarte@computer.org

***Abstract*—Software productivity is perceived by practitioners as one of the most important subjects of Software Engineering (SE), because it establishes a connection between technical and economic concerns. Nonetheless, software processes are complex and productivity means different things to different people. In order to realize the full contribution of productivity research to the practice of SE, the compilation and analysis of the diverse practitioner viewpoints and concerns is required. In this paper, we develop a systematic literature review to confirm the existence of different empirical perceptions of productivity from the distinct business sectors and knowledge areas covered in practice by SE, identifying also commonalities that may exist. This review was compiled by analyzing 73 papers on empirical studies published from 1987 to 2017. The review found great variability of study findings, particularly concerning the impacts of agile and hybrid development practices on software productivity, and research gaps that could be investigated in the future.**

***Index Terms*—Software Productivity, Systematic Literature Reviews, Empirical Studies.**

I. Introduction

Software productivity is usually perceived by practitioners as one of the most important subjects of Software Engineering (SE), because it establishes a connection between technical and economic concerns. Ever since the early studies on this subject [1], software productivity measurement considers the costs of employed personnel, equipment and third-party components as possible inputs, whereas source code, specifications and other produced software artifacts are regarded as possible outputs.

Nonetheless, this connection is not straightforward to understand, since software processes are complex *per se* [2] and there are complex interactions between process steps, such as requirements and development processes [3], between systems and software, as well as with their economics. Moreover, software productivity means different things to different people [4] and various terms denote the same productivity factors [5]. That is, the meaning of software productivity varies according to perspective and context [6]. So, in order to realize the full contribution of productivity research to the practice of SE, the compilation and analysis of the diverse practitioner viewpoints and concerns regarding software productivity is required.

In this paper, we develop a systematic literature review in order to confirm the existence of different empirical perceptions of productivity in the distinct business sectors and Knowledge Areas (KAa) covered in practice by SE, identifying also commonalities that may exist. Our review was compiled by analyzing 73 papers on empirical studies published from 1987 to 2017 in peer-reviewed journals and proceedings, whose references were recovered using the DBLP search engine [7]. Papers were classified according to the nature of their authors, covered business sectors and KAs, types and goals of reported empirical studies, as well as studied productivity metrics. These data were tabulated, classified and study findings synthesized. We hope that this review will help practitioners in addressing software productivity and in developing future research.

The paper is organized as follows: Section II describes our research methodology; Section III presents our data analyses and research findings; Section IV analyses related work and Section V discusses the threats to the validity of our research. We conclude the paper discussing our findings and presenting prospects for future research (Section VI).

II. Literature Review Methodology

A. Context Definitions

The objects of our study are software engineering processes and organizations wherein productivity can be addressed. The studied subjects are software professionals that conduct software processes and are affiliated to software organizations. The studied dependent variable is software productivity. We also study many independent variables that capture factors affecting software productivity.

Considering this conceptual framework, we study interventions in software processes that may have cause-effect relationships with productivity. Interventions are approaches to software productivity, which ultimately have some of the following goals (suggested in [9]): observing, analyzing, describing, understanding, predicting and acting on productivity.

In practice, software processes have observable inputs (measured through independent variables), suffer treatments (observed via dependent variables) and generate outcomes, connected by construct validity to interventions and effects. Software processes may have confounding factors, such as developer affects [10] or knowledge [11], making it impossible to distinguish effects from two treatments from each other.

B. Literature Review Definitions

The studies addressed in our systematic literature review appear in published papers containing practitioner views or industry data on software productivity. By practitioners, we mean software engineering professionals affiliated with private or government organizations (software industry).

TABLE I
CATEGORIES OF EMPIRICAL STUDIES ANALYZED.

Study Type	**Description** (adapted from [9])
Case Study	Adopts research questions, hypotheses, units of analysis, logic linking data to hypotheses and criteria for interpreting the findings. When some of these requirements are not satisfied, it is considered an *exploratory case study*.
Experiment	Adopts random assignment of treatments to subjects, large sample sizes, well-formulated hypotheses and selection of independent(s) variable(s), which is (are) randomly sampled. When all these requisites are satisfied, it is considered a *controlled experiment*; otherwise it is a *quasi-experiment*.
Simulation	Adopts either models to represent specific real situations / environments or data from real situations as a basis for setting key parameters in simulation models. When the simulation model is used to establish the goal(s) in (an) objective function(s), it is called an *optimization study*.
Survey	Proposes questions addressed to participants, through *questionnaires*, *(structured) interviews*, *online surveys*, *focus group meetings* and others.
Review	Incorporates results from previous studies in the analysis. When the subjects are papers, it is a *literature review*. When a well-defined methodology is used to collect secondary references, critically appraise results and synthesize their findings, it is called a *systematic literature review*. When the purpose is to map the distribution of objects across a conceptual structure, it is called a *systematic mapping*. When statistical analysis methods are adopted, it is regarded as a *meta-analysis*.

We deal both with primary and secondary studies in our review. Primary studies report on the details and results of scientifically investigating research objects and subjects, whereas secondary studies incorporate results from previous studies in the analysis. We only analyze here studies that contain practitioner views or industry data on software productivity.

We classify primary and secondary studies as case studies, surveys, experiments, simulations and reviews, eventually using qualifiers. The adopted classification appears in Table I. We restrict the scope of our research to empirical studies and, consequently, ignore position and vision papers.

C. Review Question Formulation

The goal of our study, formulated using the Goal Question Metric (GCM) methodology [12], is to review the software productivity literature with the purpose of synthesizing and analyzing this subject area considering the different underlying notions and definitions existing across the business sectors and KAs covered in practice by empirical SE. Consequently, we derive the following review questions from this goal:

RQ1 Which business sectors and knowledge areas are studied in connection to software productivity?
RQ2 How is productivity data collected and analyzed, based on which metrics?
RQ3 Which are the approaches to software productivity and what are their effects?
RQ4 What are the types of empirical studies performed on software productivity and which are their findings?

As usual in systematic literature reviews, these are very general review questions, which nevertheless guide us in achieving our synthesis and analysis goals.

D. Reference Search Strategy

We chose DBLP [7] as the primary source of bibliographic references for our study, since it is an open and curated tool that covers the main sources of published scientific research on SE, including publications in the ACM and IEEE Computer Society Digital Libraries.

Our search criteria were to find "productivity" in the paper title and "software" either in the paper title or in the publication title (proceedings or journal name). Since DBLP allows the formulation of search queries with implicit logical connectives, we carried out a publication search by inputting in DBLP the search string "software productivity" in order to obtain references matching both keywords.

We chose to carry out our review considering the period from 1987 to 2017 in order to observe publications in three entire decades and avoid short term search fluctuations, such as the varying number of papers reported in 2018 by the search engine. For the period between 1987 and 2017, our query, performed in 2018, returned 338 references.

Although the obtained set of references may seem to be small when contrasted to related work, we preferred to choose a more focused search string and treat the threats to validity that arise due to this decision as discussed in Section V.

E. Reference Exclusion Criteria

We excluded from our study paper references that failed to satisfy any of the following conditions:

1) Correspond to complete published peer-reviewed articles: We ignored reference books, theses, technical reports, abstracts and summaries retrieved in our search, preventing us from studying incomplete, partial and non-validated research results;
2) Correspond to journal papers, book chapters and conference/workshop papers whose contents were not later subsumed: This was required to ensure the analysis of the last published results, which sometimes appear with modified form or contents in relation to previously published versions;
3) Are strictly connected to software productivity: This was posed to avoid analyzing studies related mostly to other subjects (such as SE education and training), or experience reports which study specific subjects (such as productivity software) or methods, techniques and tools addressing software productivity as a secondary subject (such as software development environments that ensure higher productivity);

The compliance with these criteria was verified considering only paper title, authors, abstract and publication media. We only checked the subsumption of a paper by another one when the latest publication was also obtained as a result of our search. From the 338 references resulting from our initial search using DBLP, just 168 satisfied all these criteria.

F. Paper Inclusion Quality Criteria

For each selected bibliographic reference, we attempted to obtain an online version of the corresponding complete published paper, but we only had access to 100 of these papers. We read each obtained full text to ensure its compliance with the following quality validation checklist:

1) Reports at least on one empirical study;
2) Has a practitioner author or adopts industry data[1];
3) Clearly describes the adopted methodology;
4) Clearly explains studied variables;
5) Answers the study(ies) research question(s);
6) Provides a clear statement of main findings;

The analysis of the obtained papers based on these criteria reduced the scope of our review from 168 references to 67 papers to be analyzed.

G. Secondary Study Treatment

Among the 67 papers to be analyzed, eight were systematic secondary studies, such as literature reviews and meta-analyses. In order to include one such study in the scope of our review, either the preceding criteria had already been satisfied (in that case, it is a mixed primary and secondary study, such as [13]–[15]) or we were forced to read papers referenced therein to identify some complying with the posed requirements.

Taking advantage of this procedure, we adopted a backward snowballing technique [16], which consists in analyzing paper references in order to find relevant studies that had not been found using the adopted search string. We applied the snowballing technique only on the obtained systematic literature reviews and meta-analyses. Five additional papers were obtained in this way, including [17], which is cited in [2] and is also a systematic literature review. Consequently, we had to apply snowballing recursively yet again, on the references of [17], resulting in one additional publication. In total, this process produced six extra papers to be analyzed.

We ended up with 73 papers to be analyzed: 9 secondary systematic studies and 64 papers containing other study types.

H. Paper Processing and Treatment

The references and full versions of the selected papers were used in extracting the following data for our analyses:

1) Bibliographic key;
2) Year of publication;
3) Number of authors and practitioner authors;
4) (Qualified) empirical study type(s);
5) Studied business sector(s);
6) Main SE KA and KA topic(s);
7) Productivity approach ultimate goal;
8) Data source(s) and characterization(s);
9) Adopted productivity metric(s);
10) Employed analysis method(s);
11) Main finding(s);

The first three fields were extracted from each bibliographic reference. The number of practitioner authors was obtained from the authors and affiliations listed in each paper, reflecting author affiliations at the time of each publication.

[1]We consider industry data in an ample sense here, by admitting data and code from open source development projects and also from standards, so long as they are effectively used/adopted in industry.

TABLE II
KNOWLEDGE AREAS OF THE SWEBOK [18].

Acronym	**Chapter**	**Knowledge Area**
SWEBOK	Many	Software Engineering Body of Knowledge
SEP	8	Software Engineering Processes
SEMM	9	Software Engineering Models and Methods
SR	1	Software Requirements
SD	2	Software Design
SC	3	Software Construction
ST	4	Software Testing
SM	5	Software Maintenance
SCM	6	Software Configuration Management
SQ	10	Software Quality
SEM	7	Software Engineering Management
SEPP	11	Software Engineering Professional Practice

Study type and business sector, KA and productivity approach goal were all gathered by the author while reading each paper. We adopted the SWEBOK [18] KAs as a taxonomy for paper classification. The description of these KAs is presented in Table II. On the other hand, no *a priori* definition of studied business sectors was chosen. In this way, the business sectors presented here are those appearing in published papers.

The adopted data sources, productivity metrics, analysis methods and study findings were compiled by analyzing each paper in detail, considering the extensive body on knowledge on empirical methods found in the literature (cf. [19]).

III. Data Analysis and Research Findings

We report here our attempts to answer the review questions by analyzing primary studies and non-systematic reviews. In Section IV, we analyze systematic secondary studies.

A. Study Types, Business Sectors and KAs (RQ1)

We present the historical breakdown of the number of studied papers through the KAs of SE in Fig. 1. In the figure, we note a growth trend in the number of studied papers on software productivity and some diversification in addressed KAs. Indeed, in the five years period ending in 2017, the total number of papers doubled in relation to the period ending in 2007. Moreover, general studies were substituted by specific ones, particularly on SE management practices (cf. SEM: [4], [20]–[31]). In past periods, papers used to address traditional phases of development processes, from software design to maintenance (cf. SD, SC, ST, SQ, SQ, SCM and SM), recorded under the tag SWEBOK in our review when many were studied in a paper. Despite the general growth, we have observed just a few papers addressing social aspects (cf. SEPP) and early stages of software development (cf. SR).

The most frequent business sectors mentioned in primary studies were: software (in 18,8% of the papers); information technology (10,9%); banking and space (6,3% each); commerce (4,7%); defense, automotive and services (3,1% each). Surprisingly, 31,3% of the papers did not make explicit the target economic sectors of the respective development processes, whereas 10,9% of the papers addressed many different sectors.

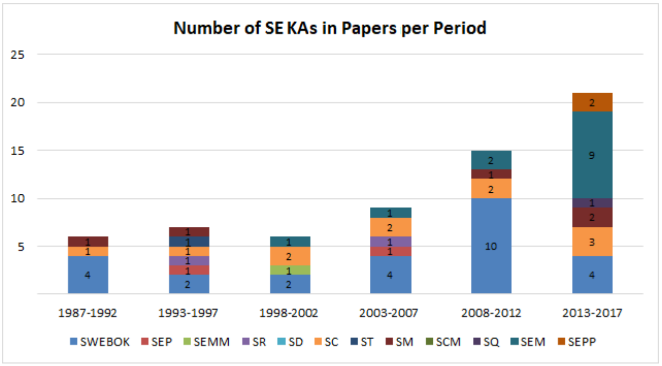

Fig. 1. Evolution of SE KAs in papers over time.

It is important to mention that the reviewed literature recognizes a significant influence of business sectors on software development productivity [32]–[34]. Moreover, in particular sectors, specific significant productivity factors were identified, such as risk classification in the banking sector [35].

B. Data Collection, Measurement and Analysis (RQ2)

Here, we are interested in analyzing how qualitative and quantitative data were collected, measured and analyzed.

Concerning data collection and data sources, we wish to understand what and how much data were collected, from which sources, by whom and in which period. Data collection time spans were determined by research convenience or according to customer needs. Sample sizes varied substantially between studies, from small samples (ex. 16 projects in [36]) to large ones (357 companies in [37] and 1000 developer pairs in [38]). Data sources were, in general, one company ([3], [4], [23], [35], [38]–[45]), many companies ([22], [46]) or publically accessible databases (CSBSG in [47]; COMPUSTAT in [48]; Experience in [32], [33]; and ISBSG in [29], [49], [50]). Data were obtained by researchers [51], [52] or self-collected by practitioners in manual [53], [54] or automated ways (e.g., by using source code management tools [20], [28], [30], [38], [55]). It is challenging to develop quantitative analyses of these aspects, given the varying amount of detail in papers. Nevertheless, although requiring extensive reprocessing – due to ambiguities, missing values and imbalanced datasets [29] – efforts to standardize data definition and collection in public databases have been considered relevant and welcome.

We also wish to understand the formulation of productivity metrics and how they are used in studies for software productivity analysis. Concerning productivity metrics, a list extracted from the studied papers is presented in Table III.

Metrics of software construction and maintenance can always be expressed as ratios between inputs and outputs of software processes [13], although some authors prefer a more algebraic formulation, using regression equations [48], [61], [62] or data envelopes [40], [49], [52], [63]. Although single ratio metrics ease data collection and analyses, factors such as elapsed time are not explicitly incorporated in these analyses [15]. Moreover, analyses based on such metrics suffer from validity threats that are not always easy to counter [75].

From the point of view of measured objects, metrics based just on source code capture only the productivity of programming and maintenance tasks [73], while others – such as analysis and design – demand measuring more structured objects, such as models [44], use case or function points (respectively [51] and [23], [26], [29], [32]–[34], [36], [43], [50], [64], [65]). Even these metrics usually ignore non-functional development and other practices, such as reuse [2].

An additional degree of complexity in measurement has been introduced by the desire to understand collaborative and distributed development, which adopt elapsed time or frequency based metrics ([28] and [20], [30], [38]). With the departure from general studies covering the whole development process and technical aspects, productivity metrics also diverged from artifact measurement. Contemporary metrics have been devised to consider diverse constructs and factors, such as affects or self-assessments ([10] and [24]).

Concerning analysis methods, the most frequently adopted in primary studies were: statistical charts (in 29,7% of the papers); descriptive statistics (21,9%); one-way ANOVA (9,4%); data envelopment analysis (7,8%); the Cobb-Douglass Model, correlational and quantitative analysis, and stepwise regression (6,3% each); linear regression, Spearman's rank correlation, Student's t-tests and system dynamics simulation models (4,7% each). It is interesting to notice that some of these methods have their roots in other disciplines, such as the Cobb-Douglass Model (which is frequently used in Econometry) and System Dynamics Models (which were developed to understand industrial processes). On the one hand, the use of analysis methods from other disciplines provides additional evidence of the maturation of software productivity measurement, but, on the other, suggests that standard methods have not been entirely effective in approaching this subject.

A plethora of other statistical tests and methods were also adopted in studies. While it is a good practice to choose the tests and methods that best fit the problem under analysis, we notice that often the preconditions for these applications are not discussed in published papers. For example, random sample selection, frequency distribution, missing data, homoscedasticity, colinearity, statistical power, effect size and goodness of fit have not always been addressed in publications. Paradigmatic examples of treatment appear in [14], [15]. Such a methodological weakness partially diminishes reader confidence in studies and should be addressed in future work.

C. Software Productivity Approaches (RQ3)

Now, we wish to elicit from studied papers the ultimate goals of software productivity approaches, with the corresponding effects. The list of analyzed papers is classified according to study goals in Table IV.

Our classification of ultimate goals embodies a notion of subsumption of less demanding goals by more stringent ones. That is why, although almost normally distributed around understanding goals, the number of papers in the table is

TABLE III
SOFTWARE PRODUCTIVITY METRICS IN PRIMARY STUDIES.

Name	**Definition**	**Count**	**Primary Studies** (in publication order, with appearence period)
many	many different metrics were used	7	[4], [5], [31], [56]–[59] (1991-2017)
TFP	total factor productivity	1	[60] (2012-2012)
EVA/y	economic value added per year	2	[48], [61] (2009-2011)
IYNR/employees	labor productivity (in year net revenue / employee number)	2	[37], [62] (2013-2017)
US$ Cost/LOC	American Dollar cost per line of code	1	[53] (1999-1999)
SDE	stochastic data envelopment (f(FP, SLOC) / person-hour)	4	[40], [49], [52], [63] (1991-2006)
adjusted size/total effort	deliverables size-effort / total-effort month	2	[14], [15] (2004-2017)
effort/task	source lines of task code / task person-hours	1	[44] (2014-2014)
FP/p-(m;d;h)	function points / person-(months;days;hours)	5	[26] (2014-2014); [23] (2003-2003); [32], [33], [50] (2000-2009)
FP/y	function points per year	1	[64] (1993-1993)
UFP/(m;h)	unadjusted function point per (month;hour)	3	[34], [43] (1999-2017); [65] (2004-2004)
UCP/p-h	use case points / person-hours	1	[51] (2017-2017)
SLOC/p-(y;m;d;h)	source lines of code / person-(years;months;days;hours)	10	[42]; [25], [39], [66]; [35]; [13], [27], [45], [47], [67]*
NCSLOC/p-(m;d)	non-comm. source lines of code / person-(months;days)	2	[54] (1994-1994); [68] (2001-2001)
DSLOC/p-(m;h)	delivered lines of code / person-(months;hours)	3	[46], [69] (1996-2005); [70] (1996-1996)
added SLOC/d	added source lines of code / days elapsed	1	[55] (2016-2016)
p-h/FP	person-hours per function point	2	[29], [36] (2011-2012)
resolution time/task	resolution time per task	1	[21] (2013-2013)
features/w	features per week	1	[41] (1996-1996)
time to first CCR	time to first contributor commit	1	[28] (2017-2017)
CCR/(month;week)	contributor commits per (month;week)	1	[38] (2009-2009); [20] (2009-2009)
inter CCR time	time between contributor commits	1	[30] (2016-2016)
SAP	self-assessed productivity	2	[10], [24] (2015-2017)
qualitative	only qualitative metrics were used	8	[3], [6], [11], [22], [71]–[74] (1991-2017)
TOTAL		64	* periods: (1999-1999); (1988-2014); (2012-2012); (1987-2009).

TABLE IV
SOFTWARE PRODUCTIVITY ULTIMATE GOALS IN PRIMARY STUDIES.

Name / Appearance	**Definition** (based on [9])	**Count**	**Primary Studies** (in order of publication)
observation (—)	Empirical observation of the objects and subjects of study (since very little is known about them).	0	—
analysis (2009-2015)	Adoption of established procedures to investigate research objects and subjects.	4	[11], [48], [61], [73]
description (1996-2017)	Provision of logical descriptions and classifications of studied objects and subjects based on analyses.	8	[5], [6], [10], [22], [34], [59], [70], [72]
understanding (1988-2017)	Explanation of why and how something happens in relation to research objects and subjects (including measurement).	34	[3], [4], [14], [15], [21], [23]–[25], [28]–[30], [32], [33], [35]–[39], [42], [43], [47], [50], [54]–[56], [58], [60], [62], [64]–[68], [74]
prediction (1991-2017)	Description of what will happen regarding studied objects and subjects.	8	[26], [40], [46], [49], [51], [52], [63], [69]
action (1987-2016)	Prescription or description of interactions with research objects and subjects so as to give rise to observable effects.	10	[13], [20], [27], [31], [41], [44], [45], [53], [57], [71]
TOTAL		64	

skewed towards actions, since these presume the completion of prediction, understanding and other less demanding goals.

One could expect that most research regarding software productivity would be concerning understanding (and measurement) goals, but this is an oversimplification. While, on the one hand, actionable theories, such as optimization models [31], provide guidance for interference in software processes, on the other hand structural equation modeling is used to describe intangible aspects of software development [59].

Although understanding goals were listed in 55% of the papers published in the two decades ending in 2017, there is growing interest in analysis and description goals, since their proportion reached 25% of the papers in the latter decade. We believe this evolution captures the emergence of new SE subjects, which often require initial productivity assessments.

D. Study Types and Reported Findings (RQ4)

Finally, we report on study types and synthesize their extent, findings and lessons learned. Due to space constraints, we focus just on the findings that contribute to analyzing our research question, by listing the relevant studies in Table V.

There is some variability in the study types reported in analyzed papers. Experiments and case studies have been dominant, respectively with at least 33% and 11% of the papers in each five years period. However, in the past decade, we noticed a substantial increase in surveys, reaching 33% of the papers in this period. This seems to mirror the emergence of new SE subjects. Indeed, four studies addressed SE management issues, such as daily practices [24], workflows [4], teamwork [22] and global development [26], whereas two other had to do with SE practices, such as feelings [10] and social practices

TABLE V
STUDY TYPES AND SELECTED FINDINGS IN PRIMARY STUDIES.

Key	Year	Auth./ Pract	Study Type	SE KA	Main findings (related to productivity)
[63]	1994	2 / 0	experiment	SM	There are significant economies of scale in maintenance projects. There may be potential gains in maintenance productivity by grouping smaller modification projects into larger planned releases.
[43]	1999	2 / 1	case study	SEMM	The adoption of OOD/OOP results in greater productivity and efficiency when compared to other approaches.
[23]	2003	4 / 1	online survey	SEM	Larger projects are more productive and have lower defects levels. Early prototyping brings the promise of subsequent work on features most valued by customers, with a positive impact on productivity.
[45]	2007	2 / 1	quasi-experiment	SWEBOK	There is no significant difference in productivity between projects developed using OOA/ODD and SA/SD, nor upon the selection of programming language. Small projects are slightly more productive than other projects. Software productivity is significantly different for distinct application domains.
[47]	2008	3 / 2	exploratory case study	SWEBOK	Project size, type and business sector are factors that influence software productivity with varying levels of significance
[36]	2011	5 / 0	case study	SC	There is a significant and positive productivity difference in Scrum-RUP projects when contrasted to traditional development.
[6]	2013	4 / 0	case study	SC	Agile team management is the most influential factor in achieving higher team productivity. Others are team size, diversity, skill, collocation and time allocation. Teams should also be aware of the negative impact of turnover.
[74]	2014	3 / 2	case study	SC	There is a decrease in the number of mistakes and interruptions in software projects, which are now coming from authorized persons. Short interaction cycles prevent endless developments. There is a significant positive impact of the agile development method Scrum on software productivity, not at the expense of software quality. This does not contribute to increasing customer satisfaction.
[30]	2016	3 / 0	experiment	SEM	The productivity of open source development decreases as the team grows in size. Due to the overhead of required coordination, open source projects are examples of diseconomies of scale.
[22]	2017	2 / 0	questionnaires	SEM	Factors that significantly affect agile team productivity are external factors and dependencies, team management and effectiveness, motivation, skillfulness and culture.
[24]	2017	5 / 1	online surveys	SEM	Factors that influence productivity are highly individual. Self-perceived productivity follows habitual patterns. Developer's work during each day is highly fragmented.

[59], with an addition generic paper on job definitions [11].

A lot of variability exists in the findings of analyzed papers. For example, while [43] mentions great improvements in software productivity due to the use of OOA/OOD/OOP, this gain is not confirmed in [45]. The effect of project and task size on productivity is reported to be positive by [63] and [23] in large (maintenance) projects, positive in OOA/OOD projects by [47], with varying impact by [45] and negative by [30] (which studies only open source projects).

There are some KAs for which significant impacts on productivity are reported due to the adoption of specific practices. The most notable example comes from agile/lean and hybrid development process practices [6]: for example, Scrum [22], [74] and Scrum-RUP [36] are reported to have definite positive impacts on productivity.

IV. RELATED WORK

The orthogonal and longitudinal characters of our study, as well as its primary concern with industry data and practitioner viewpoints, are the distinctive characteristics of our work. A synthesis of the related work is presented in Table VI.

The practitioner focus of our study led us to set specific goals and choose a distinguished methodology when contrasted to related work. Among the papers mentioned in Table VI, only one has a practitioner as a co-author [17]. Moreover, just some differentiate academic and laboratory studies from those performed in industrial settings [75], [77], [79], although studies from both sources are analyzed in each case. On the other hand, our paper selection criteria were defined taking this focus into account. In addition, we were led us to adopt a review methodology slightly different from related work, in that no concerns with publication media [75], [80] nor attempts to rate studies [75], [77] were considered necessary here, given the assumed industrial relevance of the analyzed papers.

Software productivity was reviewed herein considering the evolution of this subject over time. This is not a novelty *per se*, as it can be noticed in the tables and graphs in [17], [76], [79], [80], but this has not been presented together with KA, study type and goal breakdowns. This approach allowed us to develop historical trend analyses of software productivity research, as reported in Section III.

Our study provides an overview of software productivity with intra and inter KA analyses, whereas related work usually focuses on specific KAs. Even literature reviews are organized in this way, such as [77], [79] (which report negative impacts of test-driven development – TDD), [76] (on Scrum) and [9] (on reuse). This perspective allowed us to identify gaps in productivity research on specific KAs, as reported in Section III-A, which could be investigated in the future.

V. THREATS TO VALIDITY

A. Construct and Internal Validity

Systematic reviews are often structured in terms of subjective conceptual structures or taxonomies. The lack of sufficient definition or ambiguities in the adopted notions lead to construct validity threats. We have mitigated this kind of threat by selecting authoritative taxonomies whenever available. That is why we chose to use the SWEBOK KAs [18], as well as study types and productivity approaches adapted from [9].

In literature reviews, bias in selecting publications to be reviewed may threat construct validity. In our case, we have

TABLE VI
RELATED SYSTEMATIC LITERATURE REVIEWS.

Key	Year	Auth./ Pract.	Study Type	SE KA/ Topics	Ultimate Goal	Search Period Start	Search Period End	Queried Sources	Reference Processing (and sample sizes)	Main findings (related to productivity)
[9]	2007	2 / 0	systematic literature review	SC / -	action	1994	2005	ACM Digital Library, IEEE Explore.	After duplicate removal, 17 references were filtered and 13 selected. This list was cross-checked by manually querying the adopted sources. After detailed reading, 11 papers were analyzed.	There is significant evidence of apparent productivity gains in small and medium-scale studies. Results for actual productivity are inconsistent. The definition of productivity metrics is problematic and great variance is observed.
[17]	2008	2 / 1	systematic literature review	SWEBOK / -	action	1970	2007	ACM Digital Library, Google Scholar, IEEE Xplore, Science Direct.	53 references were filtered and 26 papers analyzed.	Communication efforts are positive for productivity, which is affected by business domains. A list of technical and soft productivity factors is distilled.
[76]	2010	5 / 0	systematic literature review	SC / XP	understanding	2000	2009	ACM Digital Library, Compendex, IEEE Xplore, Science Direct, Scopus.	274 references were obtained, 28 texts filtered and analyzed.	Scrum is related to the productivity of software projects. Other performance factors supposedly related to Scrum are mapped.
[75]	2011	1 / 0	systematic mapping	SWEBOK / -	prediction	1985	2009	ACM Digital Library, Compendex, IEEE Explore, Inspec, ISI Web of Science.	962 references were obtained, 586 were filtered and 94 selected. After detailed reading, 38 papers were analyzed.	Simple ratio measures are misleading and should be evaluated with care. SDE analysis is more robust for comparing projects. Managers should be aware of validity threats regarding productivity and should address them.
[2]	2013	3 / 0	systematic literature review	SEPP / -	understanding	1993	2003	ACM Digital Library, IEEE Xplore, ISI Web of Science, Science Direct, Taylor, Francis and Wiley Online.	187 references were obtained, 177 considered unique and 51 filtered. After detailed reading, 3 articles were selected. The list was completed by snowballing their references and 3 additional texts were included, resulting in 6 analyzed papers.	Productivity measures at job levels requiring advanced technical knowledge and skills focus either on units of a product (SLOC/Time) or planned project units (Tasks Completed/Time). There is no clear differentiation of productivity at specific job description levels.
[77]	2013	2 / 0	meta-analysis	SWEBOK / TDD	action	2002	2011	ACM Digital Library, IEEE Xplore, ISI Web of Science, Science Direct, Springer Link, Scopus.	53 references were obtained and 27 papers analyzed.	Test Driven Development (TDD) has little effect on productivity. Sub-group analyses show that the productivity drop is much larger in industries that adopt TDD, due to the additional overhead.
[78]	2015	3 / 0	systematic literature review	SC / XP	understanding	2000	2014	ACM Digital Library, IEEE Xplore, Science Direct, Springer Link.	150 references were obtained, 12 papers were filtered and analyzed.	Productivity measures are not capable of satisfying the requirements of agile development processes. They must also consider the knowledge dimension.
[79]	2016	3 / 0	systematic literature review	SC / TDD	action	1999	2014	ACM Digital Library, CiteSeerx, IEEE Xplore, Science Direct, Wiley Online Library.	1107 references were obtained, 964 considered unique and 64 filtered. After detailed reading, 24 articles were selected. This list was completed by snowballing their references, resulting in 27 analyzed papers.	There is a decrease in productivity when Test Driven Development (TDD) is adopted in industry, when compared to Test Last Development.
[80]	2017	4 / 0	systematic literature review	SWEBOK / -	understanding	1982	2015	Scopus, the Web of Science.	695 references were obtained, 625 considered unique and 224 selected. After detailed reading, 71 papers were analyzed.	Productivity metrics are usually defined using time or effort as inputs and LOC as the output, in a single ratio quantitative approach. This is explained by the fact that such measures are easier to obtain, but riskier.

used DBLP [7] as a neutral source of references, which is an open and curated tool. Moreover, review findings are determined by analyzing selected papers. The standard way to avoid analysis threats is to follow a definite research methodology and review protocol. In our study, the guidelines suggested in [8] were followed. Another source of threat is researcher bias, which we attempted to mitigate by double checking each and every one of our procedures and findings. An initiative to join our review with similar studies of some other authors is also under way.

Literature reviews are based on primary and sometimes secondary studies that may present themselves researcher biases. In order to reduce the influence of researcher opinions in our study, we decided to select only papers that were published in peer-reviewed journals and proceedings, thereby having third party validation. In addition, literature reviews suffer from possible publication biases, which correspond to the likelihood that positive results appear more frequently than negative results in published papers. However, concerning software productivity, this does not seem to be the case, given the high variability of the reported study findings.

Economic confounding factors sensibly affect software productivity. The SWEBOK mentions economic friction (everything keeping markets from having perfect competition), ecosystems and outsourcing/offshoring as examples [18]. Indeed, distance from customers, cost of delivery, restrictive regulations and/or uneven information dissemination are factors that negatively affect software costs and impact productivity. On the other hand, software development ecosystems and the existence of frameworks for intellectual property protection are likely cost savers. Offshoring and outsourcing have an ambiguous influence on software productivity. Despite the recognition of these factors impact on costs and productivity, we have seldom found, in the reviewed literature, studies of these factors. This raises concerns regarding the external validity of analyzed study findings and points out that more extensive research on productivity factors is needed.

B. External Validity

External validity threats in literature reviews arise from the existence of relevant undetected papers. Indeed, [8] alerts that no single search can find all relevant studies. In our case, we adopted DBLP [7] as a source of bibliographic references, since it covers the most relevant sources on Software Engineering, and recursively applied backwards snowballing as a second pass technique to identify additional references. This process returned 138 references, 100 of which were analyzed and thereby 73 papers selected for inclusion in our study. This sample size is larger than those mentioned in Table VI.

VI. Concluding Remarks

In this paper, we have confirmed the existence of different empirical perceptions of software productivity in the distinct business sectors and KAs covered in practice by SE, as presented in Section III-A. We also found out that there are commonalities in addressing software productivity in these KAs and sectors, mostly due to the adopted analysis methods and their respective metrics (cf. Section III-B).

The distinctive characteristics of our work were the choice of a practitioner focus and the decision to analyze only empirical studies in our research. This design appeared to be adequate because, in general, the outcomes of productivity studies are relatively distinct in practitioner/industrial contexts [59], [77] and the respective empirical studies may assume a high degree of relevance, since the studied settings are not artificial and software developers are professionals [9].

Our review found great variability of study findings, particularly concerning the impacts of agile and hybrid development practices on software productivity (cf. positive for Scrum – Section III-D – and negative for TDD – Section IV). Moreover, although we argued at the beginning of the paper that software productivity is important because it establishes a connection between technological and economic aspects, we determined here that there are more participants in this relationship, such as social and managerial factors (SEM and SEPP, described in Section III-A), as foreseen in [5], [58], with potential effects on software effort, quality and lead-time [38].

Some methodological lessons and recommendations can be derived from our work. Great care should be taken in the formulation of search strings for systematic literature reviews. While the adoption of many alternative search keys may return a nearly intractable number of references, extremely narrow search criteria (or even mistakes in formulating queries, such as in attempts to use logical connectors, which are treated by DBLP as additional search keys) may leave out important references that should be analyzed. Moreover, we noted that the use of SE taxonomies is sometimes very challenging. In particular, our decision to adopt the SWEBOK KAs [18] led our classification efforts to be very time consuming, because contemporary SE subjects are marginally addressed therein, such as those elicited in Sections III-C and III-D: agile/lean practices, web development techniques, service-oriented architectures (eg. SAAS), global development and others. In future efforts to update the SWEBOK contents, the inclusion of more practice-oriented guidance should be considered, as a way to ensure and facilitate practitioner adoption.

The strengths/trends and weaknesses/gaps in analyzed studies suggest further research. A more holistic approach in software productivity studies is needed [5], covering more or unabridged KAs (eg. SR and SEMM), sectors (noticeably industry and retail) and productivity factors (economic friction and ecosystems), while treating the lack of standardization and sufficient reporting in studies. The development of more confirmatory, replication and multi-company studies is required [73]. Efforts on diversifying data sources and establishing databases for productivity information collection and dissemination (cf. Section III-B) are also welcome. The historical and expected evolution of this field over the years – with a potential increase in the number of published studies – would provide continued evidence that the quest for productivity is still active in SE. Only with authoritative practice-oriented industrial scale studies, this quest may stride forward.

References

[1] B. W. Boehm, *Software Engineering Economics*. Prentice Hall, 1981.

[2] A. Hernández-López, R. C. Palacios, and Á. García-Crespo, "Software engineering job productivity - a systematic review," *International Journal of Software Engineering and Knowledge Engineering*, vol. 23, no. 3, p. 387, 2013.

[3] D. Damian and J. Chisan, "An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality and risk management," *IEEE Transactions on Software Engineering*, vol. 32, no. 7, pp. 433–453, 2006.

[4] A. Meyer, T. Zimmermann, and T. Fritz, "Characterizing software developers by perceptions of productivity," in *Proc. Int. Symposium on Empirical Software Engineering and Measurement (ESEM 2017)*, A. Bener, B. Turhan, and S. Biffl, Eds. IEEE, Nov. 2017, pp. 105–110.

[5] L. Cheikhi, R. E. Al-Qutaish, and A. Idri, "Software productivity: Harmonization in ISO/IEEE software engineering standards," *Journal of Software*, vol. 7, no. 2, pp. 462–470, 2012.

[6] C. de O. Melo, D. S. Cruzes, F. Kon, and R. Conradi, "Interpretative case studies on agile team productivity and management," *Information and Software Technology*, vol. 55, no. 2, pp. 412–427, 2013.

[7] M. Ley, "DBLP: Some lessons learned," *Proc. of the VLDB Endowment*, vol. 2, no. 2, pp. 1493–1500, Aug. 2009.

[8] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.

[9] P. Mohagheghi and R. Conradi, "Quality, productivity and economic benefits of software reuse: a review of industrial studies," *Empirical Software Engineering*, vol. 12, no. 5, pp. 471–516, 2007.

[10] D. Graziotin, X. Wang, and P. Abrahamsson, "Do feelings matter? On the correlation of affects and the self-assessed productivity in software engineering," *Journal of Software: Evolution and Process*, vol. 27, no. 7, pp. 467–487, 2015.

[11] A. Hernández-López, R. C. Palacios, P. Soto-Acosta, and C. Casado-Lumbreras, "Productivity measurement in software engineering: A study of the inputs and the outputs," *IJITSA*, vol. 8, no. 1, pp. 46–68, 2015.

[12] V. Basili, G. Caldiera, and H. D. Rombach, "Goal question metric (GCM) approach," in *Enciclopedia of Software Engineering*. Wiley, 2002.

[13] B. W. Boehm, "Improving software productivity," *IEEE Computer*, vol. 20, no. 9, pp. 43–57, 1987.

[14] O. Dieste, A. M. Aranda, F. U. Uyaguari, B. Turhan, A. Tosun, D. Fucci, M. Oivo, and N. Juristo, "Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2457–2542, 2017.

[15] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," *IEEE Transactions on Software Engineering*, vol. 30, no. 12, pp. 1023–1035, 2004.

[16] S. Jalali and C. Wohlin, "Systematic literature studies: Database searches vs. backward snowballing," in *Proc. ACM-IEEE Int. Symposium on Empirical Software Engineering and Measurement (ESEM 2012), Lund, Sweden*, P. Runeson *et al.*, Eds. ACM, 2012, pp. 29–38.

[17] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," in *Proc. 2nd Int. Workshop on Software Productivity Analysis and Cost Estimation (SPACE 2008), Beijing, China*, 2008.

[18] P. Bourque and R. E. Fairley, Eds., *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, 3rd ed. IEEE Press, 2014.

[19] C. Wohlin, P. Runeson, M. Høst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Verlag, 2012.

[20] P. J. Adams, A. Capiluppi, and C. Boldyreff, "Coordination and productivity issues in free software: The role of Brooks' Law," in *25th IEEE Int. Conference on Software Maintenance (ICSM 2009)*. IEEE, 2009, pp. 319–328.

[21] M. Cataldo and J. D. Herbsleb, "Coordination breakdowns and their impact on development productivity and software failures," *IEEE Transactions on Software Engineering*, vol. 39, no. 3, pp. 343–360, 2013.

[22] I. Fatema and K. Sakib, "Factors influencing productivity of agile software development teamwork: A qualitative system dynamics approach," in *Proc. 24th Asia-Pacific Software Engineering Conference (APSEC 2017)*, J. Lv, H. J. Zhang, M. Hinchey, and X. Liu, Eds. IEEE, 2017, pp. 737–742.

[23] A. MacCormack, C. F. Kemerer, M. A. Cusumano, and B. Crandall, "Trade-offs between productivity and quality in selecting software development practices," *IEEE Software*, vol. 20, no. 5, pp. 78–85, 2003.

[24] A. Meyer, L. Barton, G. C. Murphy, T. Zimmermann, and T. Fritz, "The work life of developers: Activities, switches and perceived productivity," *IEEE Transactions on Software Engineering*, vol. 43, pp. 1178–1193, Dec. 2017.

[25] R. Moazeni, D. Link, C. Chen, and B. W. Boehm, "Software domains in incremental development productivity decline," in *Proc. Int. Conference on Software and Systems Process (ICSSP 2014), Nanjing, China*, H. Zhang, L. Huang, and I. Richardson, Eds. ACM, 2014, pp. 75–83.

[26] R. C. Palacios, C. Casado-Lumbreras, P. Soto-Acosta, F. J. García-Peñalvo, and E. Tovar, "Project managers in global software development teams: a study of the effects on productivity and performance," *Software Quality Journal*, vol. 22, no. 1, pp. 3–19, 2014.

[27] N. Ramasubbu, M. Cataldo, R. K. Balan, and J. D. Herbsleb, "Configuring global software teams: a multi-company analysis of project productivity, quality, and profits," in *Proc. 33rd Int. Conference on Software Engineering (ICSE 2011), Waikiki, Honolulu, HI, USA*, R. N. Taylor, H. C. Gall, and N. Medvidovic, Eds. ACM, 2011, pp. 261–270.

[28] A. Rastogi, S. Thummalapenta, T. Zimmermann, N. Nagappan, and J. Czerwonka, "Ramp-up journey of new hires: Do strategic practices of software companies influence productivity?" in *Proc. 10th Innovations in Software Engineering Conference (ISEC 2017), Jaipur, India*, R. P. Gorthi *et al.*, Eds. ACM, 2017, pp. 107–111.

[29] D. Rodríguez-García, M. Sicilia, E. G. Barriocanal, and R. Harrison, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software*, vol. 85, no. 3, pp. 562–570, 2012.

[30] I. Scholtes, P. Mavrodiev, and F. Schweitzer, "From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in open source software projects," *Empirical Software Engineering*, vol. 21, no. 2, pp. 642–683, 2016.

[31] C. Stylianou and A. S. Andreou, "Investigating the impact of developer productivity, task interdependence type and communication overhead in a multi-objective optimization approach for software project planning," *Advances in Engineering Software*, vol. 98, pp. 79–96, 2016.

[32] K. D. Maxwell and P. Forselius, "Benchmarking software-development productivity," *IEEE Software*, vol. 17, no. 1, pp. 80–88, 2000.

[33] R. Premraj, M. J. Shepperd, B. A. Kitchenham, and P. Forselius, "An empirical analysis of software productivity over time," in *Proc. 11th IEEE Int. Symposium on Software Metrics (METRICS 2005)*. IEEE, 2005, pp. 37–46.

[34] M. Tsunoda and S. Amasaki, "On software productivity analysis with propensity score matching," in *Proc. ACM/IEEE Int. Symposium on Empirical Software Engineering and Measurement (ESEM 2017)*, A. Bener, B. Turhan, and S. Biffl, Eds. IEEE, 2017, pp. 436–441.

[35] R. Lagerström, L. M. von Würtemberg, H. Holm, and O. Luczak, "Identifying factors affecting software development cost and productivity," *Software Quality Journal*, vol. 20, no. 2, pp. 395–417, 2012.

[36] W. C. de Souza Carvalho, P. F. Rosa, M. dos Santos Soares, M. A. T. da Cunha Jr., and L. C. Buiatte, "A comparative analysis of the agile and traditional software development processes productivity," in *Proc. 30th Int. Conference of the Chilean Computer Science Society (SCCC 2011)*. IEEE, 2011, pp. 74–82.

[37] C. H. C. Duarte, "Productivity paradoxes revisited: Assessing the relationship between quality maturity levels and labor productivity in Brazilian software companies," *Empirical Software Engineering*, vol. 22, no. 2, pp. 818–847, 2017.

[38] A. Mockus, "Succession: Measuring transfer of code and developer productivity," in *Proc. 31st Int. IEEE Conference on Software Engineering (ICSE 2009)*. IEEE, 2009, pp. 67–77.

[39] A. S. Duncan, "Software development productivity tools and metrics," in *Proc. 10th Int. Conference on Software Engineering (ICSE 1988)*, T. C. Nam, L. E. Druffel, and B. Meyer, Eds. IEEE, 1988, pp. 41–48.

[40] R. D. Banker and R. J. Kauffman, "Reuse and productivity in integrated computer-aided software engineering: An empirical study," *MIS Quarterly*, vol. 15, no. 3, pp. 375–401, 1991.

[41] T. Bruckhaus, N. H. Madhavji, J. Henshaw, and I. Janssen, "The impact of tools on software productivity," *IEEE Software*, vol. 13, no. 5, pp. 29–38, Sep. 1996.

[42] T. E. Potok, M. A. Vouk, and A. Rindos, "Productivity analysis of object-oriented software development in a commercial environment," *Software Practice and Experience*, vol. 29, no. 10, pp. 833–847, 1999.

[43] D. Port and M. McArthur, "A study of productivity and efficiency for object-oriented methods and languages," in *Proc. 6th Asia-Pacific Software Engineering Conference (APSEC 1999)*. IEEE, 1999, pp. 128–135.

[44] D. Kamma and S. Kumar, "Effect of model-based software development on productivity of enhancement tasks - an industrial study," in *Proc. 21st Asia-Pacific Software Engineering Conference (APSEC 2014)*, S. S. Cha, Y. Guéhéneuc, and G. Kwon, Eds., vol. 1. IEEE, 2014, pp. 71–77.

[45] M. F. Siok and J. Tian, "Empirical study of embedded software quality and productivity," in *Proc. 10th IEEE Int. Symposium on High Assurance Systems Engineering (HASE 2007)*. IEEE, 2007, pp. 313–320.

[46] T. K. Abdel-Hamid, "The slippery path to productivity improvement," *IEEE Software*, vol. 13, no. 4, pp. 43–52, Jul. 1996.

[47] H. Wang, H. Wang, and H. Zhang, "Software productivity analysis with CSBSG data set," in *Proc. Int. Conference on Computer Science and Software Engineering (CSSE 2008)*, vol. 2. IEEE, 2008, pp. 587–593.

[48] C. Ge and K. Huang, "Productivity differences and catch-up effects among software as a service firms: A stochastic frontier approach," in *Proc. Int. Conference on Information Systems (ICIS 2011), Shanghai, China*, D. F. Galletta and T. Liang, Eds. Association for Information Systems, 2011.

[49] M. Asmild, J. C. Paradi, and A. Kulkarni, "Using data envelopment analysis in software development productivity measurement," *Software Process: Improvement and Practice*, vol. 11, no. 6, pp. 561–572, 2006.

[50] M. Tsunoda, A. Monden, H. Yadohisa, N. Kikuchi, and K. Matsumoto, "Software development productivity of Japanese enterprise applications," *Information Technology and Management*, vol. 10, no. 4, pp. 193–205, 2009.

[51] M. Azzeh and A. B. Nassif, "Analyzing the relationship between project productivity and environment factors in the use case points method," *Journal of Software: Evolution and Process*, vol. 29–53, no. 9, 2017.

[52] R. D. Banker, S. M. Datar, and C. F. Kemerer, "Model to evaluate variables impact in the productivity of software maintenance projects," *Management Science*, vol. 37, no. 1, pp. 1–18, 1991.

[53] B. W. Boehm, "Managing software productivity and reuse," *IEEE Computer*, vol. 32, no. 9, pp. 111–113, 1999.

[54] W. C. Lim, "Effects of reuse on quality, productivity, and economics," *IEEE Software*, vol. 11, no. 5, pp. 23–30, 1994.

[55] S. Bibi, A. Ampatzoglou, and I. Stamelos, "A Bayesian belief network for modeling open source software maintenance productivity," in *Proc. Int. Conference 12th IFIP WG 2.13 Open Source Systems: Integrating Communities (OSS 2016), Gothenburg, Sweden*, ser. IFIP Advances in Information and Communication Technology, K. Crowston *et al.*, Eds., vol. 472. Springer, 2016, pp. 32–44.

[56] W. Scacchi, "Understanding software productivity: Towards a knowledge-based approach," *International Journal of Software Engineering and Knowledge Engineering*, vol. 1, no. 3, pp. 293–321, 1991.

[57] S. R. Faulk, E. Loh, M. L. V. de Vanter, S. Squires, and L. G. Votta, "Scientific computing's productivity gridlock: How software engineering can help," *Computing in Science and Engineering*, vol. 11, no. 6, pp. 30–39, 2009.

[58] A. Trendowicz and J. Münch, "Factors influencing software development productivity: State-of-the-art and industrial experiences," *Advances in Computers*, vol. 77, pp. 185–241, 2009.

[59] M. Yilmaz, R. V. O'Connor, and P. Clarke, "Effective social productivity measurements during software development - an empirical study," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 3, pp. 457–490, 2016.

[60] Y. Wang, C. Zhang, G. Chen, and Y. Shi, "Empirical research on the total factor productivity of Chinese software companies," in *Proc. Int. Joint Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT 2012)*, vol. 3. IEEE, 2012, pp. 25–29.

[61] K. Huang and M. Wang, "Firm-level productivity analysis for software as a service companies," in *Proc. Int. Conference on Information Systems (ICIS 2009), Phoenix, Arizona, USA*, J. F. N. Jr. and W. L. Currie, Eds. Association for Information Systems, 2009, pp. 1–17.

[62] K. Tanihana and T. Noda, "Empirical study of the relation between open source software use and productivity of Japan's information service industries," in *Proc. 9th IFIP WG 2.13 Int. Conference on Open Source Software: Quality Verification (OSS 2013), Koper-Capodistria, Slovenia*, 2013, pp. 18–29.

[63] R. D. Banker and S. Slaughter, "Project size and software maintenance productivity: Empirical evidence on economies of scale in software maintenance," in *Proc. 15th Int. Conference on Information Systems, Vancouver, British Columbia, Canada*, J. I. DeGross, S. L. Huff, and M. Munro, Eds. Association for Information Systems, 1994, pp. 279–289.

[64] H. A. Rubin, "Software process maturity: Measuring its impact on productivity and quality," in *Proc. 15th Int. Conference on Software Engineering (ICSE 1993)*. IEEE, Apr. 1993, pp. 468–476.

[65] A. S. Parrish, R. K. Smith, D. P. Hale, and J. E. Hale, "A field study of developer pairs: Productivity impacts and implications," *IEEE Software*, vol. 21, no. 5, pp. 76–79, 2004.

[66] K. Maxwell, L. V. Wassenhove, and S. Dutta, "Software development productivity of European space, military and industrial applications," *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 706–718, 1996.

[67] T. Tan, Q. Li, B. Boehm, Y. Yang, M. Hei, and R. Moazeni, "Productivity trends in incremental and iterative software development," in *Proc. 3rd Int. Symposium on Empirical Software Engineering and Measurement (ESEM 2009)*. IEEE, 2009, pp. 1–10.

[68] W. B. Frakes and G. Succi, "An industrial study of reuse, quality, and productivity," *Journal of Systems and Software*, vol. 57, no. 2, pp. 99–106, 2001.

[69] P. Sentas, L. Angelis, I. Stamelos, and G. L. Bleris, "Software productivity and effort prediction with ordinal regression," *Information and Software Technology*, vol. 47, no. 1, pp. 17–29, 2005.

[70] D. W. Sova and C. S. Smidts, "Increasing testing productivity and software quality: A comparison of software testing methodologies within NASA," *Empirical Software Engineering*, vol. 1, no. 2, pp. 165–188, 1996.

[71] L. Kemayel, A. Mili, and I. Ouederni, "Controllable factors for programmer productivity: A statistical study," *Journal of Systems and Software*, vol. 16, no. 2, pp. 151–163, 1991.

[72] G. C. Green, A. R. Hevner, and R. W. Collins, "The impacts of quality and productivity perceptions on the use of software process improvement innovations," *Information and Software Technology*, vol. 47, no. 8, pp. 543–553, 2005.

[73] A. Hernández-López, R. C. Palacios, Á. García-Crespo, and F. Cabezas-Isla, "Software engineering productivity: Concepts, issues and challenges," *International Journal of Information Technologies and Systems Approach*, vol. 2, no. 1, pp. 37–47, 2011.

[74] K. Kautz, T. H. Johansen, and A. Uldahl, "The perceived impact of the agile development and project management method Scrum on information systems and software development productivity," *Australasian Journal of Information Systems*, vol. 18, no. 3, pp. 303–315, 2014.

[75] K. Petersen, "Measuring and predicting software productivity," *Information and Software Technology*, vol. 53, no. 4, pp. 317–343, Apr. 2011.

[76] E. S. F. Cardozo, J. B. F. A. Neto, A. Barza, A. C. C. França, and F. Q. B. da Silva, "Scrum and productivity in software projects: A systematic literature review," in *Proc. 14th Int. Conference on Evaluation and Assessment in Software Engineering (EASE 2010), Keele University, UK*, ser. Workshops in Computing, M. Turner and M. Niazi, Eds. BCS, 2010.

[77] Y. Rafique and V. B. Misic, "The effects of test-driven development on external quality and productivity: A meta-analysis," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 835–856, 2013.

[78] S. M. A. Shah, E. Papatheocharous, and J. Nyfjord, "Measuring productivity in agile software development process: a scoping study," in *Proc. Int. Conference on Software and System Process (ICSSP 2015), Tallinn, Estonia*, D. Pfahl *et al.*, Eds. ACM, 2015, pp. 102–106.

[79] W. Bissi, A. G. S. S. Neto, and M. C. F. P. Emer, "The effects of test-driven development on internal quality, external quality and productivity: A systematic review," *Information & Software Technology*, vol. 74, pp. 45–54, 2016.

[80] E. C. C. de Oliveira, D. Viana, M. Cristo, and T. Conte, "How have software engineering researchers been measuring software productivity? A systematic mapping study," in *Pro. 19th Int. Conference on Enterprise Information Systems (ICEIS 2017), Porto, Portugal*, S. Hammoudi, M. Smialek, O. Camp, and J. Filipe, Eds., vol. 2. SciTePress, 2017, pp. 76–87.

Acknowledgments

The assumptions, views and opinions in this paper are solely those of the author and do not necessarily reflect the official policy, strategy or position of any Brazilian government entity. The author wishes to thank the anonymous reviewers for their comments on a draft of this paper.