software

MDPI

Systematic Review

Software Productivity in Practice: A Systematic Mapping Study

Carlos Henrique C. Duarte [1,2]

[1] Brazilian Development Bank (BNDES), Avenida República do Chile 100, Rio de Janeiro 20031-917, Brazil; cduarte@bndes.gov.br
[2] Brazilian Institute of Geography and Statistics (IBGE), Avenida República do Chile 500, Rio de Janeiro 20031-170, Brazil; carlos.duarte@ibge.gov.br

Abstract: Practitioners perceive software productivity as one of the most important subjects of software engineering (SE) because it connects technical to social and economic aspects. Nonetheless, software processes are complex and productivity means different things to different people. In order to realize the full contribution of software productivity research to the industrial practice of SE, the analysis and synthesis of existing practitioner viewpoints and concerns are required. A systematic mapping study is developed here to investigate the existence of diverse empirical perceptions of productivity within the distinct business sectors and knowledge areas covered by the industrial practice of SE, also identifying the commonalities among them. This study adopts the DBLP and Scopus search engines to identify bibliographic references from 1987 to 2021 related to software productivity. References that do not correspond to complete not-later-subsumed articles published in peer-reviewed journals and proceedings are excluded from the analyses. Only papers reporting on empirical studies based on software industry data or that present industry practitioner viewpoints are included in these analyses. In total, 99 papers are analyzed. The mapping found great variability in study findings, particularly concerning the impacts of agile development practices on software productivity. The systematic mapping also drew methodological recommendations to help industry practitioners address this subject and develop further research.

Keywords: software productivity; GRADE; systematic mapping studies; empirical studies; software engineering

Citation: Duarte, C.H.C. Software Productivity in Practice: A Systematic Mapping Study. *Software* **2022**, *1*, 164–214. https://doi.org/10.3390/software1020008

Academic Editor: Tommi Mikkonen

Received: 1 February 2022
Accepted: 25 April 2022
Published: 6 May 2022

1. Introduction

Practitioners perceive software productivity as one of the most important subjects of software engineering (SE), because it connects technical to economic aspects. 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. However, recent studies point out that the concerns of industry practitioners regarding software productivity go far beyond technical and economic aspects and also embrace social aspects [2,3], such as affects [4], daily practices [5], teamwork [6] and job definitions [7].

Nonetheless, software productivity is not straightforward to understand, since software processes are complex per se [8] and there are complex interactions between process steps, such as requirements engineering and software design [9], and between systems and software. Moreover, software productivity means different things to different people [10] and various terms are used to denote the same productivity factors [11]. Consequently, the meaning of software productivity varies according to perspective and context [12]. In order to realize the full contribution of software productivity research to the industrial practice of SE, it is necessary to analyze and synthesize the existing practitioner viewpoints and concerns.

This paper develops a systematic mapping study to investigate the existence of diverse empirical perceptions of productivity within the distinct business sectors and knowledge areas (KAs) covered by the industrial practice of SE, also identifying the commonalities that exist among them. This study is a replication, refinement and extension of an earlier systematic literature review considering a different time frame and methodology [13]. It is noticeable that, since then, relevant studies have been published that significantly impact review findings. Moreover, by revisiting the original research goals of the systematic literature review (producing a broad overview of the subject area, providing research evidence and quantifying this evidence) and adopting an enhanced methodology, the present research effort can be characterized as a systematic mapping study, according to the criteria suggested in [14].

The present study was developed considering the recommendations formulated by Kitchenham and Charters in [14] and the PRISMA methodological guidelines [15] (which prescribe the adoption of the GRADE system [16]). This study adopted the DBLP and Scopus search engines to identify bibliographic references related to software productivity from 1987 to 2021. A total of 99 papers published in peer-reviewed journals and proceedings were analyzed reporting on empirical studies. Papers were classified according to their authors' attributes, covered business sectors and KAs, types and goals of reported studies, as well as studied productivity measures. These data were tabulated and study findings analyzed and synthesized.

The distinctive characteristics of the reported research derive from the decision to analyze only empirical studies conducted with software industry data or presenting industry practitioner viewpoints. This design appeared to be adequate because the main goal of the present study is to contribute to the SE industrial practice and, in general, the outcomes of productivity studies are relatively distinct in industrial settings [3,17]. In these environments, empirical studies assume a high degree of relevance since studied settings are not artificial and software developers are professionals [18]. This study is also original and important because it identifies the historical evolution of the software productivity field, contributes to the body of evidence and draws recommendations to help industry practitioners in addressing this subject and developing further research.

This paper is organized as follows: Section 2 describes the research methodology; Sections 3 and 4 present the analyses of primary and systematic indirect (secondary and tertiary) studies, respectively; Section 5 presents the findings and recommendations of the present study; and Section 6 discusses the existing validity threats. The last section presents some prospects for future research (Section 7).

2. Systematic Mapping Methodology

This section describes the adopted literature review protocol and systematic mapping methodology, which follow the guidelines presented in [14,15].

2.1. Context Definitions

The focus of the present study is the dependent variable of software productivity. The objects of this 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 with software organizations. Independent variables that capture factors affecting software productivity are also studied, although they are not the strict focus of the present research.

Interventions in software processes that may have cause–effect relationships with productivity are investigated here. Interventions are approaches to software productivity that have the following ultimate goals (as suggested in [18]): observing, analyzing, describing, understanding, predicting and acting on productivity.

In practice, software processes have inputs (observed through independent variables), may receive interventions that result in outcomes, and produce outputs (observed via dependent variables). Outcomes and outputs are connected to interventions and inputs

through construct validity. Depending on the studied context, software productivity may have diverse confounding factors, such as developer affects [4] or knowledge [7], making it impossible to distinguish the effects of two interventions from each other.

2.2. Systematic Mapping Definitions

Industry practitioners are SE professionals affiliated with private or public administration organizations (software industry). They are essentially distinct from academic practitioners, affiliated with universities and research centers, which are not studied here.

The empirical studies addressed in this systematic mapping are detailed in published papers. The systematic mapping deals both with primary and indirect studies. Primary studies report on scientifically investigating research objects and subjects, whereas indirect studies incorporate results from previous studies in the analysis. Primary and indirect studies are classified as case studies, experiments, simulations, surveys and reviews, eventually using qualifiers. Table 1 presents a detailed definition of this classification.

Table 1. Categories of Empirical Studies Analyzed (adapted from [18]).

Study Type	Description
Case Study	Adopts research questions, hypotheses, units of analysis, logic linking data to hypotheses and multiple criteria for interpreting the findings. If some of these requirements are not satisfied, it is considered an *exploratory case study* . It is called a *case-control* study if comparisons are drawn between a focus group and a control group, which has not suffered any intervention.
Experiment	Adopts random assignment(s) of interventions in subjects, large sample sizes, well-formulated hypotheses and the selection of (an) independent variable(s), which is (are) (randomly) sampled. If all these requirements are satisfied, it is considered a *controlled experiment*; otherwise, it is a *quasi-experiment*.
Simulation	Adopts models to represent specific real situations/environments or data from real situations as a basis for setting key parameters in models. If the model is used to establish the goal(s) of (an) objective function(s), it is called an *optimization model*.
Survey	Proposes questions addressed to participants through *questionnaires*, *(structured) interviews*, *online surveys*, *focus group meetings* and others. Participants may also be approached in a *census process* or according to *random sampling*.
Review	Incorporates results from previous studies in the analysis. If the subjects are papers, it corresponds to a *literature review*. If a well-defined methodology is used to collect references, critically appraise results and synthesize their findings, it is called a *systematic literature review*. If the purpose is to provide a broad overview of a subject area, mapping the distribution of objects across a conceptual structure, it is called a *systematic mapping*. If statistical analysis methods are adopted, it is regarded as a *meta-analysis*.

2.3. Review Question Formulation

The goal of the reported research, formulated according to the Goal Question Metric (GCM) methodology [19], is to study the software productivity literature with the purpose of analyzing and synthesizing this subject area, taking into account the diverse underlying notions and definitions that exist across the business sectors and KAs covered in industrial practice by empirical SE. The following research questions are derived 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 measures?

RQ3 Which are the approaches to software productivity and what are their effects?

RQ4 What kinds of empirical studies are developed regarding software productivity and what are their findings?

As usual in systematic mapping studies, these are general questions that help achieve the established analysis and synthesis goals. In particular, these questions point out the need to access the frequency of occurrence of business sectors, KAs, measures and goals in the studied papers. These data are correlated here with study types and findings, facilitating the compilation of temporal and demographic derived data.

2.4. Bibliographic Reference Search Strategy

DBLP (dblp.org, accessed on 24 April 2022) [20] has been used as the main tool to obtain bibliographic references for this study, since it is an open and curated tool that covers most of the sources of published scientific research on SE, including publications in the ACM and the IEEE Computer Society Digital Libraries.

The originally adopted 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 conjunctive connectives, the previous review was produced using the search string "software productivity" to obtain all references matching both keywords. However, the most recent DBLP queries missed a few previously recovered references. The root cause of this lack of repeatability was the change implemented by some bibliographic reference suppliers in the presentation style of journal names. Instead of exporting full journal names to DBLP, some publishers now only export abbreviated names. Consequently, the search string has been modified accordingly to "softw productivity".

The present study was carried out considering the period 1987–2021 to update the previous study and observe software productivity publications for 35 years. The DBLP query, last performed on 8 January 2021, returned 445 references for this period. The Scopus database (www.scopus.com) was also used as an additional source of bibliographic references. A query on Scopus last performed on the same date resulted in 489 references, but only 182 of these references were not present in the DBLP search result. Inspecting this result, it became clear that Scopus provides additional coverage of regional events and journals not directly connected to SE. However, the respective publications often would not satisfy the adopted exclusion and inclusion criteria. Consequently, just the additional references corresponding to international events and journals directly connected to SE were considered in the present study. The search on Scopus resulted in 50 additional bibliographic references to be analyzed.

Although the obtained set of references may seem small when contrasted to related work, it appears to represent the respective universe adequately since a generic search string and an international publication coverage were adopted. The threats to validity that arise from these settings were treated in the ways discussed in Section 6.

2.5. Reference Exclusion Criteria

The present study excluded from the analysis the references that failed to satisfy any of the following conditions:

1. Correspond to complete articles written in English published in peer-reviewed journals and event proceedings: The retrieved references were ignored if they corresponded to books, theses, technical reports, editorials, abstracts and summaries, preventing the analysis of incomplete, partial or not completely validated research results. The few references corresponding to papers written in other languages were also ignored;
2. Correspond to journal papers, book chapters and conference/workshop papers which were not later subsumed: Each retrieved reference was excluded if it was later subsumed by a subsequent publication. Subsumption was chosen as an exclusion criteria to avoid analyzing results that later on appear in modified form or with different contents in relation to previously published versions;
3. Are strictly connected to software productivity: This criterion was posed to avoid analyzing studies related primarily to other subjects (such as SE education and training), or experience reports that study specific subjects (such as productivity software) or methods, techniques and tools addressing software productivity as a secondary subject (such as management techniques and software development environments that ensure higher productivity);

The author of the present study verified compliance with these criteria considering only any information on paper title, authors, abstract and publication media available

online. The subsumption of a paper by another one was checked only when both references were obtained as a result of the bibliographic search. From the 495 references resulting from the initial search, only 242 satisfied all these criteria.

2.6. Paper Inclusion Criteria

The author attempted to obtain a complete version of each published paper, but only 163 of these papers were readily available online matching the selected bibliographic references. Each obtained paper was read to ensure its compliance with the following inclusion criteria:

1. Reports at least on one empirical study;
2. Has a industry practitioner author or analyses software industry data (data from the software industry is admitted here in an ample sense, covering raw data and source code from private and public administration organizations, from open databases or closed development projects, so long as they are effectively used/adopted in industry);
3. Describes the adopted methodology;
4. Explains the studied variables and measures;
5. Answers the study(ies) research question(s);
6. Provides a statement of the main findings.

In particular, the requirement that papers report on at least one empirical study prevented the inclusion of articles with opinionative content, such as position and vision papers and expert opinion texts.

Clearly, although the chosen exclusion and inclusion criteria are objective, their enforcement was based solely on the author's judgment. This poses a relevant validity threat to the findings of the present work, which is discussed in Section 6. Nevertheless, the requirement of compliance of the obtained papers with the inclusion criteria above reduced the scope of this study from 242 references to 97 articles to be analyzed.

2.7. Secondary and Tertiary Study Treatment

Among the 97 papers initially included in the analysis, there were indirect studies that analyze the findings of other articles. In order to include one such paper in the present study, the preceding exclusion and inclusion criteria had to be satisfied (in which case, it corresponds to a mixed study, presenting both a primary and an indirect study) or at least one paper referenced therein was required to comply with the criteria above. Indeed, some of the initially included papers have a mixed nature, such as [21–24], whereas nine papers present systematic indirect studies, such as literature reviews and meta-analyses ([8,17,18,25–30]).

A backward snowballing process was performed [31] to take advantage of the required inspection procedure. This technique analyzes the papers referenced in a publication to find relevant studies that had not been discovered using the adopted search strategy. The snowballing technique was applied only to the identified systematic literature reviews, systematic mappings and meta-analyses. Nine additional references were obtained in this way, including [2], cited in [8], and a systematic literature review. Consequently, snowballing was applied recursively yet again on the references of [2], resulting in one additional publication to be analyzed [32]. After verifying inclusion criteria, the backward snowballing process only produced these two extra papers to be analyzed.

Consequently, the selection process resulted in 99 papers to be analyzed in the present study: ten systematic reviews, mappings and meta-analyses and 89 articles that contain other study types. Table 2 presents a summary of the paper selection process.

Table 2. Summary of Paper Selection Processes.

STUDY	Literature Review	Systematic Mapping
Period	1987–2017	1987–2021
Primary Paper Search		
Recovered bibliographic references (a)	338	495
Excluded references after screening (b)	170	242
Papers that were not available (c)	68	90
Papers that did not meet inclusion criteria (d)	31	66
Number of included papers (e = a − b − c − d)	69	97
Backward Snowballing Search		
Recovered bibliographic references (f)	16	9
Excluded references after screening (g)	3	2
Papers that were not available (h)	8	5
Papers that did not meet inclusion criteria (i)	1	0
Number of included papers (j = f − g − h − i)	4	2
Number of Analyzed Papers (k = e + j)	73	99

The reader should not be surprised by the effectiveness reduction of the application of the backward snowballing technique in the present study. This happened due to the adoption of Scopus as an additional bibliographic reference source here. It is also important to mention that the replication of the previous study, considering a different time frame and a slightly modified publication search strategy, makes the results reported in this paper not directly comparable to those previously reported. For example, in the present case, 13 additional references were recovered by the most recent DBLP query between 1987 and 2017. Another 21 papers published in the same period are now available to the author. Moreover, one paper recovered in the previous study has been subsumed. Nevertheless, it is important to present the two studies in comparison to demonstrate the transparency of the adopted procedures in both cases.

2.8. Paper Processing and Treatment

The references and full versions of the selected papers were used to extract the following tabular data:

1. Bibliographic key;
2. Year of publication;
3. Total number of authors and industry practitioner authors;
4. Author(s) affiliation(s) information;
5. Number of studies on software productivity;
6. (Qualified) empirical study type(s);
7. Studied business sector(s);
8. Main SE KA and KA topic(s);
9. Productivity approach ultimate goal;
10. Data source(s) and their characterization(s);
11. Interventions and outcomes, if applicable;
12. Adopted productivity measure(s);
13. Employed analysis method(s);
14. Main finding(s);
15. Conflict of interest and funding information.

The first two fields were extracted from each bibliographic reference. Author affiliations, numbers of authors and reported studies on software productivity, conflicts of interests, and funding information were obtained from the data included in each paper. Study type and business sector, KAs and productivity approach goal were gathered by the author while reading each paper. The list of non-foundational SWEBOK KAs [33] was

used as a coding taxonomy for included paper subjects classification. Table 3 presents the description of these KAs. On the other hand, no a priori definition of studied business sectors was chosen, so they are reported here in the way they appear in published papers. The data sources, productivity measures, analysis methods and findings of each study were compiled by inspecting each paper in detail, considering the extensive body of empirical methods in the SE literature (cf. [34]).

Table 3. Non-Foundational Knowledge Areas of the SWEBOK (adapted from [33]).

Acronym	Chapter	Knowledge Area
SWEBOK	Many	Software Engineering Body of Knowledge
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
SEM	7	Software Engineering Management
SEP	8	Software Engineering Processes
SEMM	9	Software Engineering Models and Methods
SQ	10	Software Quality
SEPP	11	Software Engineering Professional Practice

2.9. Analysis and Synthesis Methods

The main methods used in analysis and synthesis procedures were the visual inspection of included papers and the tabular presentation of collected data. Missing data were noted and presented in this way in connection to each research question. In addition to tabular presentations, textual descriptions of collected data are also presented here, along with the respective occurrence frequencies. Frequencies corresponding to single occurrences are omitted for simplicity of presentation.

Some bar charts are also presented here to show evidence at a high level of granularity and facilitate the development of temporal trend and gap analyses of included studies. However, these charts are not comparable to those shown in the previous study [13] since a period of 35 years is analyzed here, equally divided into five-year periods from 1987 to 2021.

3. Data Analysis and Primary Study Finding Compilation

This section describes the attempts to answer the research questions by analyzing the findings and data collected in primary studies and non-systematic reviews. The findings of other indirect studies are analyzed in Section 4 since the respective papers have distinct structures and adopt different methodologies. The certainty analysis in this body of evidence and a synthesis of the findings of the present systematic mapping study are detailed in Section 5.

The demographics of the analyzed papers are as follows. Concerning authorship, 35% of the papers have industry practitioners among their authors, whereas 65% only have authors affiliated with academic institutions. In terms of gender, 48.5% of the papers have female authors, whereas 51.5% only have male authors. Figure 1 presents the geographic distribution of the authors of analyzed articles.

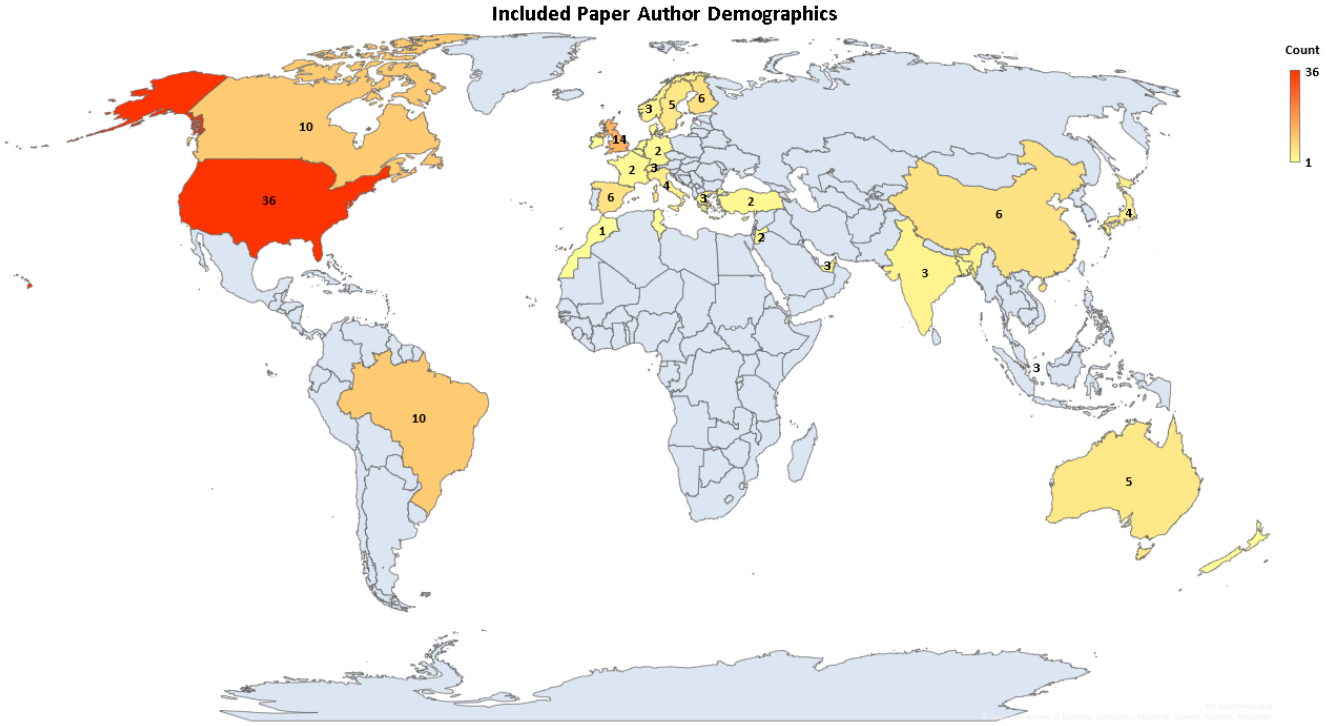

Figure 1. Country of author's affiliation(s) of included papers.

3.1. Business Sectors and KAs in Studies (RQ1)

Figure 2 presents the historical breakdown of the number of studied papers through the KAs of SE. Overall, the figure displays a growth trend in the number of studied papers on software productivity. In the last five years, the number of analyzed papers is three times greater than in the initial period. Some diversification in addressed KAs is noticeable, with general studies (recorded under the tag SWEBOK in this study when many phases were addressed in a single paper) being substituted by specific ones, mainly SE management practices (cf. SEM: [5,6,10,35–47]). In the past, papers addressed more traditional phases of development processes, from design to maintenance (cf. SD, SC, ST, SQ, SCM and SM). Despite the general growth, only a few articles address social aspects (cf. SEPP) and early stages of software development processes (cf. SR).

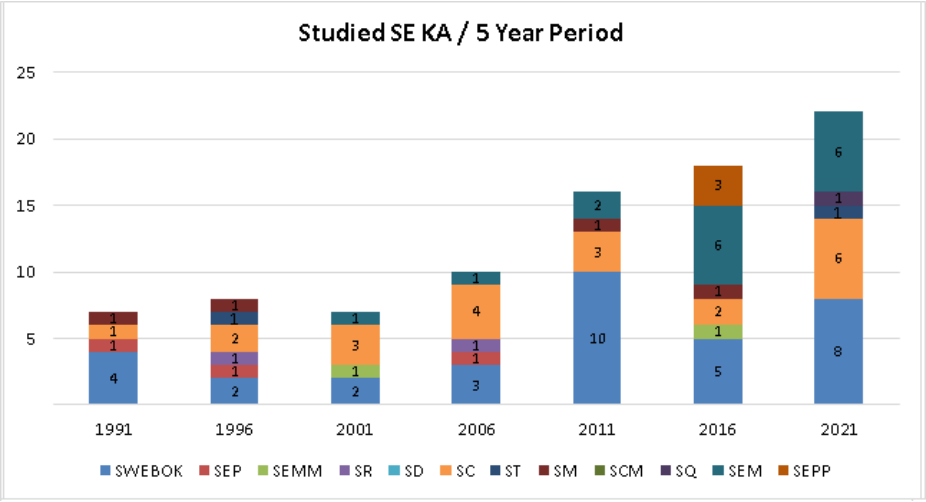

Figure 2. Evolution of SE KAs in papers over time.

The most frequent business sectors mentioned in primary studies are: the business of software development (in 23.6% of the papers); other information technology businesses (11.2%); banking, space and commerce (4.5% each); defense and services (3.4% each); and automotive, education and government (2.2% each). Surprisingly, 32.6% of the papers did not mention the target economic sectors of the studied development processes, whereas 11.2% of the papers addressed many different sectors.

It is important to mention that the reviewed literature recognizes a significant influence of business sectors on software development productivity ([24,48–56]). Moreover, in particular sectors, specific significant productivity factors were identified, such as risk assessment in the banking sector [57].

3.2. Data Collection, Measurement and Analysis (RQ2)

Concerning data sources and data collection, one challenge is understanding what and how much data were collected, from which sources, by whom and in which period. Data collection periods were determined by convenience or according to research customer needs. Sample sizes varied substantially between studies, from small samples (e.g., 16 projects in [58]) to large ones (687 companies in [54], 1000 developer pairs in [59], and 700,000 issue reports in [60]). Data sources were, in general, one organization ([3,9,10,21,23,35,37,44,45,47,48,57–59,61–76]), many organizations ([4–6,12,22,38,40,42,77–89]), publicly accessible databases (CSBSG in [52]; COMPUSTAT in [90]; Experience in [49,50,55,91]; ISBSG in [39,51,55,56,70,91–93]; SEC in [53]) and open-source software (OSS) repositories (SourceForge in [94]; GitHub in [43,95]; and the Apache Projects Directory in [60]). Data were obtained by researchers [63,79,85–87,96] or collected by practitioners, in manual [64,80] or automated ways (e.g., by using source code management tools [36,45,59,74,84,97,98]). It is not easy to analyze these aspects quantitatively, given the varying amount of detail in papers. Nevertheless, although requiring reprocessing—due to ambiguities, missing values, and imbalanced datasets [39]—efforts to standardize data definition and collection in public databases have been considered not only relevant but also welcome and should be addressed in the future.

Another challenge is understanding the formulation of productivity measures and how they are used in studies for software productivity analysis. Table 4 presents a list of productivity measures extracted from the studied papers. Often, software construction and maintenance measures are expressed as ratios between inputs and outputs of software processes [21]. However, some authors prefer a more algebraic formulation, using regression equations ([90,99,100]) or data envelopes ([62,63,70,96]). Although single ratio measures ease data collection and analyses, factors such as elapsed time are not explicitly incorporated in these analyses [22]. Moreover, analyses based on such measures suffer validity threats that are not always easy to counter [29].

From the point of view of the studied objects, measures based only on source code capture only the productivity of programming, testing and maintenance tasks [32], while others, such as systems analysis and software design, demand measuring the production of more structured artifacts—models [75], use cases [85], function points ([35,37,39,42,49,50,53,55,57,58,69,78,97]) and even formal proofs [101]. These measures usually ignore non-functional requirements and practices such as reuse [8].

An additional degree of complexity in measurement is introduced by recent efforts to understand collaborative and distributed development, which adopt elapsed time ([43,45]) or frequency-based ([36,59,95,102]) measures. With the departure from general studies covering the whole development process and technical aspects, productivity measures also diverged from software artifact measurement. Apart from new techniques, such as self-assessments ([5,47,87]), contemporary measures have been devised to consider various constructs and factors such as affects [4] and job enthusiasm [89].

Table 4. Software Productivity Measures in Primary Studies.

Name	Definition	Count	Primary Studies (with Occurrence Period)
many	many different measures were used	14	[3,7,10,11,24,32,44,46,51,83,91,103–105] *
TFP	total factor productivity	1	[106,107] (2012–2021)
EVA/y	economic value added per year	2	[90,99] (2009–2011)
labor productivity	annual net revenue/number of employees	2	[54,100] (2013–2017)
US$ Cost/LOC	American Dollar cost per line of code	1	[80] (1999–1999)
SDE	stochastic data envelopes: f(FP, SLOC)/person-hour	4	[62,63,70,96] (1991–2006)
CP/US$ cost	change points/cost in American dollars	1	[65] (1995–1995)
adjusted size/total effort	deliverables size-effort/total-effort month	2	[22,23] (2004–2017)
effort/task	source lines of task code/task person-hours	1	[75] (2021–2021)
FP/p-(m; d; h)	function points/person-(months; days; hours)	8	[42]; [35,37,57,77]; [49,50,53] **
FP/y	function points per year	1	[78] (1993–1993)
UFP/(m; h)	unadjusted function point per (month; hour)	5	[55,69] (1999–2017); [56,93,97] (2004–2020)
UCP/p-h	use case points/person-hours	2	[85,86] (2017–2018)
LOP/p-w	lines of proof/person-weeks	1	[101] (2014–2014)
SLOC/p-(y; m; h)	source lines of code/person-(years; months; hours)	11	[48]; [41,61,68,108]; [21,38,52,71,72,84] ***
NCSLOC/p-(m; d)	non-comm. source lines of code/person-(months; days)	2	[64] (1994–1994); [81] (2001–2001)
DSLOC/p-(m; h)	delivered lines of code/person-(months; hours)	3	[79,92] (1996–2005); [67] (1996–1996)
added SLOC/d	added source lines of code/days elapsed	1	[98] (2016–2016)
p-h/FP	person-hours per function point	2	[39,58] (2011–2012)
resolution time/task	resolution time per task	1	[40] (2013–2013)
features/w	features per week	1	[66] (1996–1996)
CLOC/m	committed lines of code per month	1	[94] (2010–2010)
time to first CCR	time to first contributor commit	1	[45] (2017–2017)
daily contribution	committed lines of code and files per day	1	[102] (2021–2021)
CCR/(m; w)	contributor commits per (month; week)	1	[59] (2009–2009); [36] (2009–2009)
PR/m	pull requests per month	1	[95] (2021–2021)
inter-CCR time	time between contributor commits	1	[43] (2016–2016)
SAP	self-assessed productivity	8	[4,5,47,60,74,76,87–89] (2015–2021)
qualitative	only qualitative measures were used	7	[6,9,12,73,82,109] (1991–2017)
TOTAL		89	periods: * (1991–2020). ** (2014–2014); (1991–2012); (2000–2009). *** (1999–1999); (1988–2014); (1987–2011).

Furthermore, going back to traditional productivity measurement techniques, there are also ways of accounting for software process inputs and outputs based on monetary values. These measures should be used with caution, for example due to the adoption of different currencies in studies, such as the Renminbi in [106,107], the Iene in [100] and the American Dollar in [65,80,90,99]. The possibility of changes in the purchasing power of adopted currencies due to the effects of physical or economic processes on assets, such as depreciation and inflation [54], may hinder study comparability.

The analysis methods most frequently adopted in primary studies were: descriptive statistics (39.3% of the papers); statistical charts (37.1%); linear regression (20.2%); correlation analysis and ANOVA (12.4% each); qualitative analysis methods (7.9%); data envelopment analysis (DEA) and stepwise regression (6.7% each); the Cobb–Douglas model, the least-squares method and Kruskal–Wallis tests (5.6% each); Spearman's rank correlation and Student's *t*-tests (4.5% each); Pearson's rank correlation, system dynamics simulation models and Wilcoxon Rank-Sum tests (3.4% each); analogy-based estimation, logistic regression, Mann–Whitney tests, Markov's chains, regression trees and structural equation modeling (2.2% each).

Some adopted methods have their roots in other disciplines, such as the Cobb–Douglass model and DEA (frequently used in Econometrics) and System Dynamics Models (developed to understand industrial processes). Usually, the adequacy of these methods is justified by practical reasons. For example, [96] mentions that DEA allows for the identification of productivity factors that are under managerial control and have a significant impact on productivity, and, once identified, management can take steps to retain and amplify positive factors and mitigate or eliminate negative ones. Method transference also happens in relation to computer science branches such as machine learning: analogy-based estimation [85,93], regression trees [87,91], Bayesian belief networks [98], thematic network analyses [12] and K-means learning [85] are also adopted in included studies. They are used to overcome limitations in statistical techniques, such as the requirement of normally distributed variables. The use of analytical methods from other disciplines provides additional evidence of the maturation of software productivity measurement, but suggests that traditional methods have not been entirely effective in approaching this subject.

While it is a good practice to choose the tests and methods that best fit the problem under analysis, the preconditions for their application are frequently not discussed in published papers. For example, random sample selection, missing data, frequency distribution, homoscedasticity, colinearity, goodness of fit, statistical power and effect size have not always been addressed in publications (with few exceptions, such as [22,23]). Such methodological weaknesses partially diminish confidence in some studies and, as a general concern, should be addressed in the future.

3.3. Software Productivity Approaches (RQ3)

The studied papers also identify the ultimate goals in software productivity approaches. Table 5 classifies the analyzed articles according to these goals. Figure 3 presents the historical breakdown of the number of studied papers through these goals. Between 40% and 60% of the studied papers have understanding goals, confirming the perception that software productivity has always demanded explanations as to why and how outputs and outcomes are observed in each studied context.

Table 5. Software Productivity Ultimate Goals in Primary Studies.

Name/Occurrence	Definition (Based on [18])	Count	Primary Studies (in Order of Publication)
observation (—)	Empirical observation of the objects and subjects of study (since little is known about them).	0	—
analysis (2009–2015)	Adoption of established procedures to investigate what are the research objects and subjects.	4	[7,32,90,99]
description (1996–2017)	Provision of logical descriptions and classifications of studied objects and subjects based on analyses.	8	[3,4,6,11,12,55,67,82]
understanding (1988–2021)	Explanation of why and how something happens in relation to research objects and subjects (including measurement).	49	[5,9,10,22,23,35,39–41,43,45–54,56–61,64,65,69,73,74,76–78,81,84,87,88,94,97,98,100–106,108]
prediction (1991–2021)	Description of what will happen regarding studied objects and subjects.	14	[42,62,63,68,70,79,85,86,89,91–93,95,96]
action (1987–2021)	Prescription or description of interactions with research objects and subjects so as to give rise to observable effects.	14	[21,24,36–38,44,66,71,72,75,80,83,107,109]
TOTAL		89	

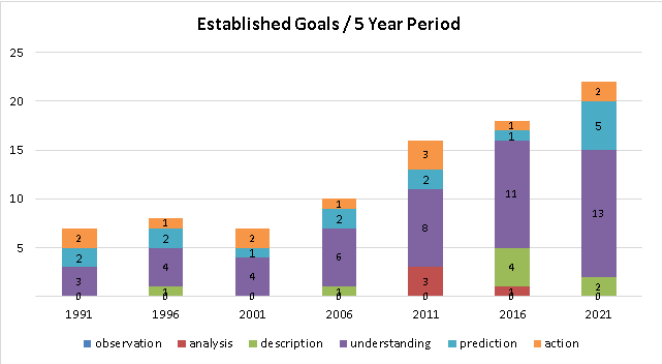

Figure 3. Evolution of ultimate goals in papers over time.

It is important to mention that the adopted classification of ultimate goals embodies a notion of subsumption of less demanding goals by more stringent ones. That is why the numbers of papers in the table and figure are slightly skewed towards action goals since they presume the fulfillment of prediction, understanding and other goals.

One might expect that most research on software productivity would have understanding (and measurement) goals, but this is an oversimplification. While, on the one hand, actionable theories, such as optimization models [44], guide interference in software processes, on the other, techniques like structural equation modeling help describe intangible aspects of software development [3].

Although understanding goals were listed in 55% of the papers published in the five years ending in 2021, there is reasonable diversification of study goals. This diversity follows the emergence of new subjects in SE, which require the formulation of distinct productivity study goals. Examples are Software as a Service (SaaS, [90,99]), standardization [11] and agile practices ([6,12]). The development of more studies with observation, analysis, description and action goals should be addressed in the future.

3.4. Study Types and Reported Findings (RQ4)

Figure 4 presents the historical breakdown of the number of study types found in analyzed papers. Therein, some variability can be noticed. Experiments and case studies were dominant, corresponding to at least 33% and 14% of the papers in each five-year period, respectively. However, there has been a substantial increase in surveys, reaching 40% of the papers in the last period. The future still holds the promise of more studies on knowledge-based simulation and optimization models of software productivity [103], which correspond to only three included papers.

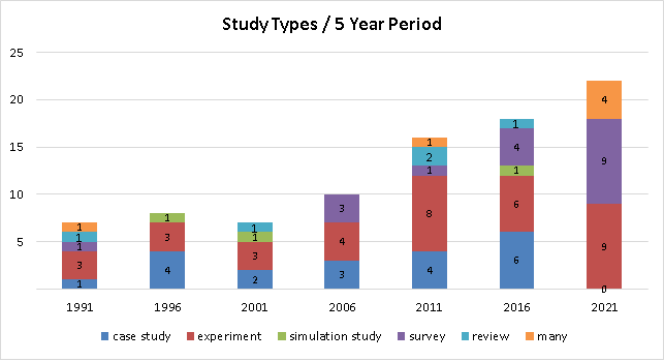

Figure 4. Evolution of study types in papers over time.

The growth in the number of surveys seems to mirror the emergence of new interests in SE in the last decade. Indeed, four studies address SE management issues, such as daily practices [5], workflows [10], teamwork [6] and global development [42], whereas two others concern human aspects, such as affects [4] and social practices [3], which are typical concerns in agile practices. It is also noticeable in the last decade that many surveys investigate the business of engineering software, such as technical debt [87], job definitions and satisfaction ([7,47]), working environments [76], and health and well-being conditions ([88,102]).

Concerning the analysis of study findings, the hierarchical structure of the non-foundational KA topics and subtopics detailed in the SWEBOK [33] is adopted here, in conjunction with some specific subtopics that do not belong to this hierarchy (OSS, reuse and SE economics databases). The KA subtopics explicitly addressed in the included papers are Software Process Improvement (SPI), Rapid Application Development (RAD), Capability Maturity Model (CMM), Object-Oriented Design (OOD) and Test-Driven Development (TDD). Included papers are grouped according to this classification scheme and their findings are analyzed in the context of each KA.

Special attention is given here to productivity factors studied in most included papers. A structured list of productivity factors appears in [2] and a definition of these factors through theoretical constructs is proposed in [24]. Herein, these definitions are taken for granted and the influence of factors on software productivity is coded considering their directionality, effect and significance whenever possible.

The direction of influence of productivity factors corresponds to positive ($\uparrow$), conditional ($\rightarrow$), indistinguishable ($\sim$) or negative ($\downarrow$) contributions. The symbols $<$ and $>$ are used to denote greater-than and lower-than relationships. When the reported results are statistically significant (p-value < 0.05) or strongly statistically significant (p-value < 0.01), the usage of these relational symbols is doubled or tripled, respectively. In order to codify the findings reported in included studies, the square symbol ($\square$) is used as a placeholder for any of the aforementioned relational symbols. Finally, when inconclusive results are reported, the symbol ? is used next to the relational symbols.

3.4.1. Studies Using SE Economics Databases

Many studies analyze the productivity data of software development projects that are contributed by private companies to public databases, as discussed in Section 3.2. These experiments and case studies cover either the entire body of knowledge on SE or only software construction. They analyze two specific subjects, not necessarily in an exclusive way: the adequacy of models and methods for software productivity measurement or prediction and specific software productivity factors. Table 6 summarizes of the respective study types and findings, together with the respective KA topics and the total numbers of authors, practitioner authors and reported studies in each paper.

Table 6. Primary Study Types and Findings that Use Public Databases.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(MaxwellF00) [49]	2/2	1	case study	SWEBOK	The factors mostly impacting software productivity are company and business sector. Companies must statistically analyze available data to develop benchmarking equations based on key productivity factors.
(PremrajSKF05) [50]	4/2	1	controlled experiment	SWEBOK	There is evidence of improved productivity over time, with variations coming from company and business sector. Insurance and commerce are the least productive, while manufacturing is the most productive sector among the studied projects. There is no significant difference in productivity between new developments and maintenance projects.
(SentasASB05) [92]	4/0	1	quasi-experiment	SWEBOK	The ability of ordinal regression models to classify any future project in one of the predefined categories is high based on the studied databases.
(AsmildPK06) [70]	3/0	1	controlled experiment	SWEBOK	It is possible to develop proper exponential statistical models to predict productivity, but linear models are inappropriate. DEA can incorporate the time factor in analyses and can be used to determine the best performers for benchmarking purposes.
(MosesFPS06) [51]	4/0	1	case study	SC	The studied company outperforms those in the ISBSG database by approximately 2.2 times. Possible explanations are that projects are lead by staff with knowledge of systems and business processes and an optimized model-based development process is adopted. The Bayesian credible intervals gives a more informative form of productivity estimation than it would be possible using the usual confidence interval alternative for the geometric mean of the ratio.
(WangWZ08) [52]	3/2	1	exploratory case study	SWEBOK	Project size, type and business sector are factors that influence software productivity with varying significance levels. There is no evidence that team size and adopted programming languages affect productivity. There is no significant difference in productivity between new developments and redevelopment projects.
(BibiSA08) [91]	3/0	2	quasi-experiment	SWEBOK	A combination of the methods of association rules and regression trees is prescribed for software productivity prediction using homogeneous datasets. Their estimates are in the form of rules that the final user can easily understand and modify.
(Tsuno09) [53]	5/1	1	quasi-experiment	SWEBOK	Architecture and team size have a strong correlation with productivity. Business sector, outsourcing and projects skewed towards the implementation ensure moderate productivity.
(GeH11) [90]	2/0	1	quasi-experiment	SWEBOK	The stochastic frontier approach takes both inefficiency and random noise into account and is a better approach for productivity analysis. It allows the understanding of SaaS company dynamics and catch-up effects by comparison to traditional companies.
(RodriguezSGH12) [39]	4/0	1	controlled experiment	SEM	Improvement projects have significantly better productivity than new development and larger teams are less productive than smaller ones.
(TsunodaA17) [55]	2/0	1	quasi-experiment	SWEBOK	The propensity score analysis can determine undiscovered productivity factors. The company business sector and the development platform are significantly related to software productivity.
(LavazzaMT18) [56]	3/0	1	quasi-experiment	SC	The adopted primary programming language has a significant effect on the productivity of new development projects. The productivity of enhancement projects appears much less dependent on programming languages. The business area and architecture have significant effect on productivity. No evidence of the impact of CASE tools usage on productivity was determined. The productivity of new development projects tends to be higher than that of enhancement projects.
(LavazzaLM20) [93]	3/1	1	quasi-experiment	SC	Software enhancement costs more than new software development, at least for projects greater than 300 Function Points. There is a lot of variability in studied data to reach this conclusion.

# of Studies = number of studies.

The included papers investigate the adoption of the following new analytical models and methods:

- Ordinal regression models [92];
- Bayesian credible intervals [51];
- Data envelopment analysis [70];
- Association rules and regression trees [91];
- Stochastic frontier approach [90];
- Propensity score matching [55].

Their findings correspond to positive results: the investigated models and methods are considered appropriate for software productivity measurement or prediction.

The included studies based on databases also analyze many different software productivity factors considering diverse contexts. These factors can be classified as organizational/managerial factors or technical factors. On the one hand, the studied organizational/managerial factors are business sector, company, level of outsourcing, project and team size. On the other, the investigated technical factors are software architecture, development platform, adopted programming language and development tools. The contexts of these studies are software development and maintenance projects.

The complete coding of the respective relationships between factors and software project productivity is presented in Appendix A.1. However, some derived relationships are shown below as a way of illustrating the coding process:

1. Level of outsourcing $\downarrow \rightarrow \uparrow$ development project productivity ([53]);
2. Development platform $\rightarrow \rightarrow$ development project productivity ([55]);
3. Adoption of development tools $\rightarrow ?$ development project productivity ([56]);
4. Business sector $\square$ software project productivity
   (where $\square = \rightarrow \rightarrow$ for [56]; $\square = \rightarrow$ for [49,52,53]).

The coding of the finding related to outsourcing should be read as "development project productivity decreases as the outsourcing level increases". Moreover, the conclusion on development platforms should be read as "development platform significantly contributes to development project productivity". In addition, the outcome related to development tools should be read as "No evidence was found that the adoption of development tools contributes to development project productivity". Futhermore, the finding related to business sectors should be read as "software (that is, maintenance and new development) project productivity is (significantly, according to [56]) affected by the business sector".

3.4.2. Other Studies Covering the SWEBOK

Among the studies covering the whole SWEBOK that do not adopt databases as a data source, it is possible to find many experiments and surveys that analyze productivity factors or software productivity from regulatory ([11]) and economic ([99,104,107]) perspectives. There are also some literature reviews ([24,103,105]) among these papers. Table 7 presents a summary of the respective study types and findings.

Papers with an economic perspective investigate study subjects in connection to economic measures. The specific topics studied in these papers are:

- Economies and diseconomies of scale in pure, mixed and non-SaaS firms ([99]);
- Positioning of software companies in supply chains as prime contractors, intermediate contractors, end-contractors, and independent enterprises ([104]);
- Regional differences in the level of development of software companies ([107]).

These studies cannot rely on standardized variables defined in public databases. So, it becomes more difficult to aggregate evidence, but the coding of each derived relationship is presented in Appendix A.2. It is possible to divide again productivity factors into organizational/managerial and technical ones. Factors in the former category are organizational structure, risk assessment, team experience with users and technology, and technical debt. In the latter category, there are UCPs, FPs, LOCs, the adoption of development platforms and programming languages, and RAD and reuse practices.

Many papers report on subjective factors influencing software productivity [79], which are even more difficult to quantify. They are technical supervision, working conditions, achievement, responsibilities and recognition [109]; motivation, performance, management, compensation and rewards, organizational climate and happiness [32]; job definitions [7]; external interruption, environment adaptation and emotional issues [88]; satisfaction with the work environment and ability to work privately [76]; job enthusiasm, peer support for creativity, and helpful feedback [89].

3.4.3. Requirements Engineering

Just two included papers study software requirements. The first half of Table 8 summarizes the respective study types and findings. Productivity factors studied therein are requirements volatility, engineer communications and management tools adoption. A synthesis of the studied productivity factor relationships appears in Appendix A.3.

3.4.4. Object-Oriented Development

Four included papers study object-oriented analysis and design in connection to software productivity. The second half of Table 8 summarizes their types and findings. The productivity factors studied therein are application domain, project size, mobility incentives, deadline enforcement and effective OOD adoption. Appendix A.4 presents a synthesis of the respective factor relationships.

3.4.5. Software Construction

Many included papers cover specific activities of software construction that refer to the creation of working software through a combination of coding, verification, unit testing, integration testing, and debugging. These activities are generally regarded as software construction in the SWEBOK [33]. The included papers report on experiments [22,23] and case studies [71,84,97] that investigate specific productivity factors, discuss the adequacy of regression models for model-based development productivity prediction [86] and self-reported productivity evaluation [74]. An additional paper proposes a new software effort measurement technique [65].

Table 9 presents a summary of the included study types and findings. Once again, studied factors can be classified as managerial/organizational and technical ones. In the former category, there are formal education, team capabilities and knowledge. In the latter, there are software architecture, requirements volatility, development tools and pair programming. Therespective relationships are synthesized in Appendix A.5.

3.4.6. Software Reuse

As already mentioned, one of the most important software construction practices is reuse. That is why included papers addressing this theme are treated separately here. Each respective article is unique in its combination of study type and analysis methods. They all address the practice of reuse as a relevant factor in software development project productivity. Table 10 presents a summary of the corresponding study types and findings. A synthesis of the relationships in individual studies is presented in Appendix A.6.

3.4.7. Open-Source Software

Open-source software is a transversal theme in the SWEBOK. The included papers on OSS are also treated separately here since the respective experiments and case studies analyze specific productivity factors and measures. Table 11 presents a summary of the individual study types and findings.

Interestingly, OSS project productivity factors and measures are substantially different from those in other KAs. Apart from OSS adoption, the studied factors have technical nature. They are software aging, adopted programming language and development increment. The respective relationships are synthesized in Appendix A.7.

3.4.8. Software Testing

Among the included papers, two study testing in connection to software productivity. The first half of Table 12 summarizes the respective study types and findings. The investigated productivity factors are testing project difficulty and task process transferability in software testing projects. A synthesis of the findings related to the factorss that influence testing productivity is presented in Appendix A.8.

3.4.9. Software Maintenance

Four included papers study maintenance in connection to software productivity. The second half of Table 12 summarizes the respective experiments and case studies.

Yet again in this case, productivity factors can be classified into the organizational/managerial and technical categories. The studied organizational/managerial factors are team capabilities, mentors succession and experience, and offshoring. The technical factors are domain knowledge, workload, development increment and maintenance granularity, as well as artifact coupling and quality control. A synthesis of the respective relationships appears in Appendix A.9.

3.4.10. Software Engineering Management

According to the SWEBOK [33], software engineering management (SEM) is defined as the application of management activities (planning, coordinating, measuring, monitoring, controlling, and reporting) to ensure that software products and services are delivered efficiently and effectively to the benefit of stakeholders. Many included papers cover these concerns through experiments, case studies, surveys and simulation studies.

Table 13 presents a summary of the included papers. The studied factors are inherently managerial or related to the management of technical activities. Inherently managerial factors are offshoring, team autonomy, mobility, experience heterogeneity and management, task coordination and completion incentives, and project size. The management of the following aspects is also studied: adoption of process models, RAD, development and testing tools. Appendix A.10 synthetizes the respective relationships.

Many papers report on the existence of social and personal factors that facilitate managing software development professionals and teams for improved productivity. These are usually measured in qualitative or self-reported ways. They are that the use of the Pareto optimal set in personnel allocation and task scheduling supports better management decisions [44]; each developer has a highly fragmented daily routine, and factors influencing their productivity are quite individual [5]; personalized recommendations for improving software developers' work are essential to optimize their productivity [10]; new hires with prior internships tend to perform better than others in the beginning, take several weeks to reach the productivity levels of experienced employees, and their team support effect decreases with time [45]; code-based metrics outperform commit-based metrics in reflecting developer perceived productivity, and triangulation can strengthen organizational confidence in productivity measures [46].

3.4.11. Rapid Application Development

Software construction and engineering management are tightly connected to the specific practices of rapid application development. They cover lean and agile practices, such as the adoption of Scrum, prototyping, Test-Driven Development and pair programming. The first half of Table 14 summarizes these surveys and case studies together with the respective findings. The studied factors are related to the managerial aspects of team management, size, diversity, turnover, as well as to personal capabilities and Scrum adoption. A synthesis of the studied relationships is presented in Appendix A.11.

Table 7. Primary Study Types and Findings Covering the SWEBOK.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(Boehm87) [21]	1/1	3	many	SWEBOK	Productivity is slightly higher in the prototyping approach since it consumes fewer resources. Regarding 4GLs, great variance is observed. The increase in demand and software costs create a need to improve productivity. There are opportunities in (a) getting the best from people; (b) making process steps more efficient; (c) eliminating steps; (d) eliminating rework; (e) building simpler products; (f) reusing components.
(Duncan88) [61]	1/1	1	case study	SWEBOK	There has been a significant increase in productivity, which is attributed to increased code reuse due to the use of software productivity tools.
(KemayelMO91) [109]	3/0	1	questionnaire-based survey	SWEBOK	A manager can determine the personnel, process, and customer factors that significantly affect productivity. The most significant personnel factors are experience with virtual machines and the user community. Among process factors, the most significant are the definition of life cycle and cost estimation, the use of modern programming languages, and the power of adopted equipment. Concerning user factors, the experience with the community, computers, and with analysts and programmers are the most significant. The most significant motivational factors are technical supervision, working conditions, achievement, responsibilities and recognition.
(Scacchi91) [103]	1/0	1	review	SWEBOK	Analytical instruments or tools are required to model and measure productivity in ways that managers and developers can employ. This may lead us away from simple quantitative measures towards knowledge-based tools that embody symbolic and qualitative (dynamic) models.
(AbdelHamid96) [79]	1/0	1	simulation study	SWEBOK	Software productivity is usually eroded by motivational factors and communication overhead. These have to do with the failure to execute perfectly reasonable management practices since most software projects are conducted with poorly defined requirements, staff turnover, volatile hardware and others.
(Maxwe96) [108]	3/0	1	controlled experiment	SWEBOK	Organizational differences are the primary source of variance in software productivity, but development team size, application types, programming languages and development tools are also essential and controllable productivity factors. This highlights the need for companies to establish their own software metrics database and benchmark their data against other companies.
(FaulkLVSV09) [83]	5/4	2	interviews, questionnaire-based survey	SWEBOK	Barriers to productivity improvement in scientific computing are the specific development approaches adopted in this domain, since they present bottlenecks that current practices cannot avoid.
(HuangW09) [99]	2/0	1	controlled experiment	SWEBOK	Mixed SaaS firms may enjoy significant economies of scale and are more efficient than pure SaaS firms. Pure SaaS companies exhibit smaller economies of scale than conventional companies and are more productive only in utilizing capital assets.
(MinetakiM09) [104]	2/0	1	survey	SWEBOK	Software enterprises are classified as prime contractors, intermediate subcontractors, end-contractors, and independent enterprises. Intermediate subcontractors are the least productive. However, those possessing highly skilled workers have high productivity levels.

Table 7. *Cont.*

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(TrendM09) [105]	2/0	1	review	SWEBOK	Successful productivity improvement depends on humans. The obtained results do not support the traditional belief that software reuse is the key to productivity improvements. Other frequent factors mentioned in the literature are tools and methods. Factors facilitating team communication and work coordination are also important in software outsourcing. Selecting the right factors is the first step towards quantitative productivity management.
(HernandezLopezPGC11) [32]	4/0	1	review	SWEBOK	Motivation, performance, management, compensation and rewards, organizational climate and happiness can influence productivity. The influence of reuse should be further studied. Another challenge is to develop measures that differentiate new developments from maintenance.
(CheikhiARI12) [11]	3/0	1	review	SWEBOK	Factors mentioned in industrial software engineering standards may affect productivity. For the ISO 9126-4 standard, productivity is a quality characteristic, whereas there are metric models in the IEEE 1045 standard to deal with productivity. Their differences do not allow building a consensual productivity model. However, the latter can be used as part of the former.
(LagerstromWHL12) [57]	4/0	1	controlled experiment	SWEBOK	Developed function points, adopted software platforms, and risk classification significantly impact software costs and productivity. Two factors often assumed to affect the project cost, the efficiency of the implementation and the costs of pre-study, failed to display significant impacts.
(Wang12) [106]	4/1	1	quasi-experiment	SWEBOK	Productivity increases come from technology adoption and progress. Education is also a factor that positively affects the productivity of software companies.
(HernandezLopezCSC15) [7]	4/0	1	interviews, questionnaire-based survey	SWEBOK	SE practitioners can be classified as knowledge workers. They perceive some SE factors both as inputs and as outputs. New productivity measures should consider job position definitions to guide developing the respective metrics.
(AzzehN17) [85]	2/0	4	randomized experiment	SWEBOK	Learning how to predict productivity from environmental factors is more efficient than using expert assumptions. Still, it is better to exclude them from calculating UCPs and make them available only for computing productivity.
(BeskerMB19) [87]	3/0	2	online survey, interviews	SWEBOK	Developers waste, on average, 23% of their time due to technical debt and they are frequently forced to introduce additional technical debt in those cases in which it is already present. The most common activity on which additional time is spent is performing further testing.
(BezerraEA20) [88]	7/0	1	online survey	SWEBOK	During the COVID-19 pandemic, 74.1% of the surveyed developers said their productivity remained good or excellent and 84.5% felt motivated and communicated easily with co-workers. The main factors influencing productivity are external interruption, environment adaptation and emotional issues.

Table 7. *Cont.*

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(ChapettaT20) [24]	2/0	2	many	SWEBOK	The structured synthesis method allows inferring the intensity and confidence of the factors affecting software development productivity. It offers an initial theoretical framework for representing the current status of empirical knowledge in software development productivity.
(JohnsonZB21) [76]	3/2	2	online survey, interviews	SWEBOK	In productivity models, the overall satisfaction with the work environment and the ability to work privately with no interruptions are as important and significant factors. Private offices were linked to higher perceived productivity across all disciplines. For software engineers, another vital factor for perceived productivity was communicating with the team and leads.
(MurphyHillEA21) [89]	9/9	4	randomized questionnaire-based survey	SWEBOK	Factors that most strongly correlate with self-rated productivity are non-technical factors, such as job enthusiasm, peer support for new ideas, and receiving helpful feedback about job performance. Compared to other knowledge workers, software developers' self-rated productivity is more strongly related to task variety and working remotely.
(ZhaoWW21) [107]	3/0	1	quasi-experiment	SWEBOK	There are regional differences in the level of development of local software companies. Different public policy promotion paths should be adopted in each case considering simultaneously all the identified gaps in the degree of higher education, the scale of enterprises and the level of investment in research and development activities and fixed assets, acting on them accordingly.

Table 8. Primary Study Types and Findings on Requirements Engineering and OO Development.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(HenshawJMB96) [66]	4/2	1	exploratory case study	SR	There is a perception of productivity improvements due to personal software processes. To increase productivity, requirement management tools should be selected considering the project size and development process.
(DamianC06) [9]	2/1	1	questionnaire-based survey	SR	RE productivity improvements arise from improved project communication and reduced rework. Basing designs and test cases on more accurate specifications provides consistent and informative direction for requirement engineers.
(PotokV97) [68]	2/1	1	simulation study	SWEBOK/OOD	The lack of incentives for early completion of intermediate project tasks and rigorous enforcement of final project deadlines may trigger delays and negatively affect software development productivity. Common business practices might lower project productivity and project completion probability. Organizations must control the productivity ranges in which their development teams operate.
(PotokVR99) [48]	3/2	1	controlled experiment	SWEBOK/OOD	The governing influence on OOD productivity may be the business workflow, but not the development approach. There is significant evidence that productivity increases as project size increases. Business deadlines may have a strong influence on the overall productivity of projects.
(PortM99) [69]	2/1	1	case study	SEMM/OO	The adoption of OOD coupled with OOP significantly improves overall project productivity and efficiency, but OO development approaches are less efficient than traditional approaches in the requirements phase.
(SiokT07) [72]	2/1	1	quasi-experiment	SWEBOK/OOD	Productivity is significantly different for distinct application domains. There is no significant difference in productivity between projects developed using OOA/ODD and SA/SD or programming language. Small projects are slightly more productive than medium and large projects.

Table 9. Primary Study Types and Findings on Software Construction.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(Chatman95) [65]	1/1	1	case study	SC	The change-point measure permits both combined and individual productivity measurement for design, implementation and test activities. It supports a conceptual approach to productivity measurement at a higher level than in each development activity.
(KitchenhamM04) [22]	2/0	1	controlled experiment	SC	A software productivity measure related to effort can be formulated when several jointly significant factors are related to effort. The practice of reuse is determined to affect productivity significantly. Executives evaluate that requirements stability, customer satisfaction and customer/staff personality type may contribute to software productivity.
(ParrishSHH04) [97]	4/0	1	case study	SC	Highly collaborative pairs are dramatically (4 times) less productive than pairs working on the same task but not simultaneously. Programming pairs can learn to work more productively together over time by devising their productive collaboration process. Any productivity gains reported with pair programming are likely due entirely to the role-based protocol rather than to any inherent consequences of working closely in pairs.
(TomaszewskiL06) [71]	2/0	1	case study	SC	The following are identified as productivity bottlenecks in software construction: unstable requirements and lack of programming tools (large); quality of platform documentation, and too optimistic planning (average). Apart from treating these bottlenecks, higher knowledge of the development language and platform and adoption of reuse practices may improve productivity.
(Tan09) [84]	6/0	1	case study	SC	The collected data present a clear trend of decreased software productivity over the years. Staff capabilities, software architecture, and other development tasks affect software productivity, either positively or negatively. In incremental development, the assumption that productivity will vary from increment to increment cannot be taken for granted.
(DiesteEtAll17) [23]	8/0	10	controlled experiment	SC	Familiarity with a unit testing framework or IDEs appears to affect software productivity positively. Years of practical or academic programming experience do not influence programmer productivity, so the routine practice does not appear to lead to improved performance. However, academic learning, which could be considered an instance of deliberate practice, influences quality and productivity.
(AzzehN18) [86]	2/0	2	controlled experiment	SC	Learning productivity ratios for each project look more reasonable and efficient than using a static ratio for all software organization projects. Using effort regression models based on UCP size variables is more accurate than effort estimation-based productivity models.
(BellerOBZ21) [74]	4/4	1	quasi-experiment	SC	A simple linear regression model could explain almost half of the variance in self-reported productivity when expressed as a product and process measure. Organizations should be aware of the large conceptual discrepancy between self-reported and measured productivity and that optimizing for individual productivity is different from optimizing for team productivity.

Table 10. Primary Study Types and Findings on Software Reuse.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(Boehm99a) [80]	1/0	1	review	SC/REUSE	Elicited reuse success factors for improved productivity are: (a) adoption of a software product line (SPL) approach; (b) business case analyses; (c) focus on black-box reuse; (d) empowerment of SPL managers; (e) establishment of reuse-oriented processes; (f) adoption of an incremental approach; (g) usage of metrics-based management; (h) establishment of an SPL strategy.
(BankerK91) [62]	2/0	1	quasi-experiment	SC/REUSE	There is an order of magnitude productivity gain due to the adoption of reuse in software construction.
(Lim94) [64]	1/1	2	case study	SC/REUSE	Performing cost–benefit analyses for potential new products helps determine which should be created or reengineered to be reusable.
(FrakesS01) [81]	2/0	1	quaisi-experiment	SC/REUSE	More reuse results in higher quality, but the relationship between the amount of reuse and productivity is unclear.

Table 11. Primary Study Types and Findings on Open-Source Software.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(AdamsCB09) [36]	3/1	1	case study	SEM/OSS	Contributor commits change over time in OSS projects. Depending on the project nature, there is an irregular ramp-up period, after which developers start increasing their productivity.
(KreinMKDE10) [94]	5/1	1	randomized experiment	SWEBOK/OSS	Programming language fragmentation is negatively related to the total amount of code contributed by developers. For a developer who programs in multiple languages, it appears that he or she is most productive when language fragmentation is minimal.
(TanihanaN13) [100]	2/0	1	case study	SWEBOK/OSS	The economic effect of the OSS segment for the labor productivity of the Japanese information service sector is positive. However, each OSS produces a variety of economic effects.
(MoazeniLCB14) [41]	4/0	1	case study	SEM/OSS	Incremental development productivity decline varies significantly according to product categories and domains.
(ScholtesMS16) [43]	3/0	1	controlled experiment	SEM/OSS	The productivity of OSS development decreases as the team grows in size. Due to the overhead of required coordination, open-source projects are examples of diseconomies of scale.
(LiaoEA21) [95]	6/0	9	many	SWEBOK/OSS	The flow of participants and the popularity of an open-source ecosystem impact its capacity to produce information. Positive communication by participants can hurt the ability of an ecosystem to solve practical problems. No matter what stage the ecosystem is in, its age will impact productivity. The number of publishers participating in ecosystems and of followers harm ecosystems' net productivity.

Table 12. Primary Study Types and Findings on Software Testing and Maintenance.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(SovaS96) [67]	2/1	2	case study	ST	There was a consistent agreement between expert opinion ratings and the developed testing productivity measure. Both determined that difficult projects have lower productivity with the adoption of a testing methodology.
(JaloteK21) [75]	2/1	3	many	ST	There are clearly identifiable differences between the task processes of high-productivity programmers and the task processes of average-productivity programmers. Task processes of high-productivity programmers were transferred to average-productivity programmers by training them on the key steps missing in their processes but commonly present in the work of their high-productivity peers. A substantial productivity gain was found among average-productivity programmers due to this transfer.
(BankerDK91) [96]	3/0	1	controlled experiment	SM	High project quality does not necessarily reduce maintenance productivity. A significant positive impact is observed on maintenance productivity by project team capabilities and good response time. A negative significant impact is identified due to the lack of previous experience in the application domain.
(BankerS94) [63]	2/0	1	quasi-experiment	SM	Project size has an important influence on maintenance productivity. There are significant economies of scale in the studied maintenance projects. There may be significant gains in maintenance productivity by grouping simple modification projects into larger planned releases.
(Mockus09) [59]	1/1	2	controlled experiment	SM	Larger projects, overload mentors and offshoring succession significantly reduce the productivity ratio. The breadth of mentor experience and succession of mentors' primary product significantly increase productivity.
(BibiAS16) [98]	3/0	1	case study	SM	Small methods produce nearly maximal productivity in the majority of cases. Tightly coupled systems exhibit low productivity rates, a negative effect of coupling on maintainability.

Table 13. Primary Study Types and Findings on Software Engineering Management.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(MacCormackKCC03) [35]	4/1	1	online survey	SEM	Larger projects are more productive and have lower defect levels than smaller ones. Early prototyping and daily builds promise subsequent work on the features most valued by customers, with a significant positive impact on productivity. Other practices are not correlated to productivity. There is danger in assuming the implementation of more flexible processes piecemeal by picking-and-choosing practices because there are complex interactions among them.
(RamasubbuCBH11) [38]	4/0	1	controlled experiment	SEM	Firms that distribute software development across long distances benefit from improved productivity. Variations in configurational characteristics of distributed teams lead to different performances. Locally tailored, agile, and interaction-oriented process models are associated with improved productivity. Project configurations that attain high productivity tend to achieve low quality and vice versa. An imbalance in the experiences of personnel significantly decreases productivity.
(Mohapatra11) [37]	1/0	1	quasi-experiment	SEM	Application complexity affects productivity negatively, and training in the application domain has an opposite effect. Productivity tends to increase with the availability of documentation and testing tools and better client support.
(CataldoH13) [40]	2/1	2	case study	SEM	Identifying the right set of relevant work dependencies and coordinating accordingly has a significant impact on increasing productivity. When developers' coordination patterns are congruent with their coordination needs, productivity increases.
(PalaciosCSGT14) [42]	5/0	1	questionnaire-based survey	SEM	Performance in global development projects is lower than in-house projects due to the lack of attention to tasks by software managers. This is due to communication, coordination and control overheads. The management of offshore projects affects their performance in negative ways.
(StylianouA16) [44]	2/0	2	optimization study	SEM	Significantly improved performance is perceived in case managers present accessibility, responsivity and neglect their superior roles. The Pareto optimal set, which is generated from models, supports managers better deciding on who will work on what and when.
(MeyerBMZF17) [5]	5/1	1	online survey	SEM	Productivity is a highly personal matter, and perceptions of what is considered to be productive are different across participants. Productivity and the factors that influence it are highly individual. The daily work of each developer is highly fragmented.
(MeyerZF17) [10]	3/1	1	online survey	SEM	Personalized recommendations for improving software developers' work are essential to optimize personal and organizational workflows. Software developers can be classified as social, lone, focused, balanced, leading or goal-oriented developers.
(RastogiT0NC17) [45]	5/4	1	quasi-experiment	SEM	New hires tend to take several weeks to reach the same productivity levels as experienced employees. The effect of team support decreases with time. Employees with prior internships tend to perform better than others in the beginning.
(OliveiraEA20) [46]	6/0	1	many	SEM	Code-based metrics outperformed commit-based metrics, reflecting team leaders' perceptions of developer productivity. Data triangulation can strengthen organizational confidence in productivity metrics.
(StoreyEA21) [47]	6/4	1	randomized online survey	SEM	The perception of existence of an engineering system, impactful work, autonomy, and capability to complete tasks positively affect self-assessed productivity. In contrast, the possibility of mobility, compensation and job characteristics affect it negatively. The relationships of these factors to job satisfaction is statistically significant in many models for different work contexts.

Table 14. Primary Study Types and Findings on Rapid Application Development and Software Practices.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(CarvalhoRSCB11) [58]	5/0	1	exploratory case-control study	SC/RAD	There is a significant and positive productivity difference in Scrum-RUP projects when contrasted to traditional development. The proposed hybrid process incorporates the advantages and benefits of the dynamics of agile principles but recognizes the importance of conducting rigorous requirements management and architecture in the traditional way.
(MeloCKC13) [12]	4/0	1	case study	SC/RAD	Agile team management is the most influential factor in achieving higher team productivity. Team size, diversity, skill, collocation and time allocation are critical factors for designing agile teams. Teams should be aware of the negative impact of member turnover.
(KautzJU14) [73]	3/2	1	case study	SC/RAD	There is a decrease in the mistakes and interruptions in software projects due to the adoption of Scrum. Short interaction cycles prevent endless developments. Scrum has a significant positive impact on software productivity, not at the expense of software quality. However, customers do not perceive these improvements.
(FatemaS17) [6]	2/0	1	interviews, questionnaire-based survey	SEM/RAD	Factors that significantly affect agile team productivity are external factors and dependencies, team management and effectiveness, motivation, skillfulness and culture.
(KuutilaMCEA21) [102]	5/0	2	online survey, interviews	SC/RAD	Using software repository variables to predict developers' well-being or productivity is challenging due to individual differences. Prediction models developed for each developer individually work better.
(GraziotinWA15) [4]	3/0	1	questionnaire-based survey	SEPP	Affects (emotions, moods and feelings) impact the cognitive activity of individuals. Valence (the attractiveness of an event) and dominance (change in the sensation of control of a situation) are positively related to self-assessed productivity. Arousal (the intensity of emotional activation) does not provide additional explanatory power to the developed model.
(MantylaADGO16) [60]	5/0	1	quasi-experiment	SEPP	Issue reports of different types produce a fair valence variation (the attractiveness of an event). Increases in issue priority typically increase arousal (the intensity of emotional activation). The resolution of an issue increases valence. As the resolution time of an issue increases, so does the individual arousal assigned to the issue.
(YilmazOC16) [3]	3/0	1	interviews, questionnaire-based survey	SEPP	Software productivity has a multi-factor structure. Productivity is highly associated with social productivity (an intangible asset related to social life, information awareness, fairness, frequent meeting, reputation, social debt, team communication and cohesion) and moderately associated with social capital (intangible resources related to group characteristics, norms, togetherness, sociability, neighborhoods, volunteerism and trust). The productivity of software development was found to be higher for smaller software teams.

3.4.12. Software Engineering Professional Practice

The Software Engineering Professional Practice (SEPP) is concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering in a professional, responsible, and ethical manner [33]. The second half of Table 14 summarizes the findings of the respective experiments and surveys.

The study findings on the professional practice of software engineering are rather qualitative. In [3], software productivity is determined to be highly associated with social productivity (an intangible asset related to social life, information awareness, fairness, frequent meeting, reputation, social debt, team communication and cohesion) and moderately associated with social capital (intangible resources related to group characteristics, norms, togetherness, sociability, neighborhoods, volunteerism and trust). In [4], social productivity is decomposed into affects (emotions, moods and feelings), valence (the attractiveness of an event), dominance (change in the sensation of control of a situation), and arousal (the intensity of emotional activation), although arousal does not provide additional explanatory power in their usage together. However, in [60], high valence, arousal and dominance are positively related to self-assessed productivity in studying the resolution of issue reports stored in source code repositories.

3.4.13. Software Processes, Quality, Models and Methods

Finally, the findings of studies covering the remaining KAs are grouped in this section. They address software engineering processes (SEP), software quality and software engineering models and methods. Table 15 summarizes the corresponding experiments and surveys together with the respective findings. A synthesis of these rather diverse studied relationships is presented in Appendix A.12.

Table 15. Primary Study Types and Findings oon SE Processes, Quality, Models and Methods.

Key	Auth./Pract.	# Studies	Qualified Study Type	SE KA	Main Findings (Related to Productivity)
(LowJ91) [77]	2/0	1	quasi-experiment	SEP	Overall, there is no statistical evidence for a productivity improvement or decline resulting from CASE tools. Close evaluation of individual projects reveals support for traditional learning-curve patterns and the importance of staff training in new technology.
(Rubin93a) [78]	1/0	1	quaisi-experiment	SEP	Among studied companies, only 20% had information on their portfolio size, only 3.3% on portfolio changes and only 2% had information about quantifiable aspects of software quality. Process improvement and organizational aspects are important factors for software productivity.
(GreenHC05) [82]	3/0	1	questionnaire-based survey	SEP	There is increased perception of productivity improvements due to personal software processes.
(Duarte17a) [54]	1/1	1	quaisi-experiment	SQ	There is no evidence of improved labor productivity or productivity growth in companies with appraised software quality levels. Companies with appraised quality maturity levels are more or less productive depending on their business nature, capital's main origin, and maintained quality level. There is statistically significant evidence that software productivity variance decreases as a company with appraised quality levels moves towards higher levels.
(StaplesEA14) [101]	6/0	1	quasi-experiment	SEMM	Lines of proof is a problematic measure, and so improved size measures are required. Effort is highly correlated with proof size. Since there are proofs that are much simpler and less complex than other proofs, it would be expected that effort and productivity depend on proof complexity. Still, empirical data do not provide support for this belief.

4. Related Work Discussion and Indirect Study Finding Compilation

Many related studies on software productivity have an indirect character, in that they provide systematic reviews and mappings covering third-party studies. Systematic indirect studies are treated separately from primary and mixed-type studies here to prevent double-counting of study results and comparing studies that adopt substantially distinct methodologies. Table 16 presents a synthesis of this related work. Interestingly, the present research is unique in the sense that it is a mixed tertiary study [110], once not only primary but also indirect studies are analyzed here.

Table 16. Included Indirect Study Types and Findings.

Key	Auth./Pract.	Study Type	SE KA/Topics	Ultimate Goal	Initial Year	Final Year	Queried Sources	Reference Processing/Paper Selection	Main Findings (Related to Productivity)
(MohagheghiC07) [18]	2/0	systematic literature review	SC/-	action	1994	2005	ACM Digital Library, IEEE Explore.	After duplicate removal, 17 references were obtained and 13 selected. After reading, 11 papers were included and analyzed.	There is significant evidence of apparent productivity gains in small and medium-scale studies. Results for actual productivity are rather inconsistent. The definition of productivity measures is problematic and great variance is observed.
(WagnerR08) [2]	2/1	systematic literature review	SWEBOK/-	action	1970	2007	ACM Digital Library, Google Scholar, IEEE Xplore, Science Direct.	962 references were obtained, 586 were filtered and 53 selected. After reading, 38 papers were included and analyzed.	Communication efforts are positive for software productivity, which is also sensible to business domains.
(CardozoNBFS10) [26]	5/0	systematic literature review	SC/RAD	understanding	2000	2009	ACM Digital Library, Compendex, IEEE Xplore, Science Direct, Scopus.	274 references were obtained, 28 papers included and analyzed.	The relationship between the adoption of Scrum and the productivity of software projects is likely positive.
(Peter11) [29]	1/0	systematic literature review and systematic mapping	SWEBOK/-	prediction	1985	2009	ACM Digital Library, Compendex, IEEE Explore, Inspec, ISI Web of Science.	53 references were obtained, 26 papers included and 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 research and address them.
(HernandezLopezPG13) [8]	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 selected. After reading, 3 articles were included. The list was completed by snowballing 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 according to specific job descriptions.
(RafiqueM13) [17]	2/0	meta-analysis	SWEBOK/TDD	action	2002	2011	ACM Digital Library, IEEE Xplore, ISI Web of Science, Science Direct, Springer Link, Scopus.	274 references were obtained and 28 papers included and analyzed.	Test-Driven Development (TDD) has little effect on productivity. Subgroup analyses show that the productivity drop is much larger in industries which adopt TDD, due to the additional overhead.
(ShahPN15) [30]	3/0	systematic literature review	SC/RAD	understanding	2000	2014	ACM Digital Library, IEEE Xplore, Science Direct, Springer Link.	150 references were obtained, 12 papers were included and analyzed.	Productivity measures are not capable of satisfying the requirements agile development processes. They must also consider the knowledge dimension.
(BissiNE16) [25]	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 selected. After reading, 24 articles were included. This list was completed by snowballing and 3 additional texts included, 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.
(OliveiraVCC17) [27]	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 reading, 71 papers were included and analyzed.	Productivity measures are usually defined using time or effort as the inputs and LOC as the output. Single ratio measures are easier to obtain, but riskier to adopt.
(OliveiraCCV18) [28]	4/0	tertiary systematic literature review	SWEBOK/-	action	–	–	ACM Digital Library, Engineering Village, IEEE Xplore Digital Library, Scopus and Web of Science.	After duplicate removal, 240 references were selected. After random sampling, 4 publications were included and analyzed.	No single classification exists for software productivity factors, but they are organized in product, process, project and people categories. The reviewed literature studies 35 influential factors over which organizations must intervene to obtain software productivity improvements.

The focus of the present study on industry data and practitioner views yields a set of specific goals and a choice of a distinguished methodology in comparison to related work. The paper selection criteria adopted here were defined taking this focus into account. Only one has a practitioner co-author [2] among the papers mentioned in Table 16. Some differentiate academic and laboratory studies from those in industrial settings ([17,25,29]), although studies from both sources are analyzed in each case. Moreover, the review methodology adopted here is slightly different from related work, in the sense that there are no specific concerns with publication media ([27,29]) nor attempts to score individual studies according to pre-established quality criteria ([17,29]), given the assumed practical or industrial relevance of each analyzed paper and the existence of some subjectivity in establishing quality criteria for paper scoring, respectively.

Software productivity is analyzed here considering the evolution of this subject over time. This is not a novelty per se, as shown in the tables and graphs in [2,25–27], but software productivity has not been analyzed elsewhere in connection to KA, study type and study goal breakdowns. This approach enables the development of historical trend analyses of software productivity research, as reported in Section 3.

The present study provides an overview of software productivity with intra- and inter-KA analyses, whereas related work usually focuses on specific KAs. In particular, some related literature reviews are organized in this way, such as [17,25] (which report minor or negative impacts of TDD on software productivity), [26,30] (report inconclusive results concerning the adoption of Scrum and agile methods) and [18] (on selectively positive impacts of reuse). On the other hand, the approach adopted here enables the identification of gaps in productivity research on specific KAs, as reported in Section 3.1, which should be investigated in future research.

In addition, the present research builds upon the general technical and methodological findings reported in the related work, particularly [2,8,27–29]. A summary of findings and a derivation of methodological recommendations based on included studies are respectively developed in Sections 5.3 and 5.4. The present study also has similarities with the mixed-type study reported in [24]. Whereas a structured synthesis method is employed in [24] to determine the level of certainty attributed to each study findings, here the GRADE system [16] is applied, as described in Section 5.2.

5. Systematic Mapping Findings and Recommendations

Systematic mappings and literature reviews evaluate individual quality of evidence and provide high certainty in a body of evidence. Following the PRISMA guidelines [15], the GRADE system [16] is adopted in this section to achieve these goals.

The GRADE system prescribes a structured approach to synthesizing evidence in literature reviews and systematic mappings. First, a risk of bias assessment is developed (Section 5.1). Next, an evaluation of certainty in the body of evidence (Section 5.2) is conducted. Finally, an evidence profile and a summary of findings table are constructed, together with a narrative discussion of the main study findings (Section 5.3). In addition, some methodological recommendations are derived here from the lessons learned in the analysis and synthesis of included paper findings (Section 5.4).

5.1. Risk of Bias Assessment

Systematic mappings and literature reviews are based on analyses of included primary and indirect studies, which may present risks of research, reporting and other biases. There is a plethora of sources of research bias that may affect the credibility of reviewed studies, such as participant selection and allocation, missing data or non-response, observations and measurements, study performance and withdrawals or exclusions. Moreover, included studies may suffer from reporting biases, which correspond to the publication or not of research findings depending on the nature and direction of results. Specifically, reporting biases can be classified in publication, selective reporting, time-lag, language, citation, multiple publication and location biases. Finally, other sources of bias come from conflicts

of interest, which occur whenever professional misjudgment or unduly influences happen due to secondary interests, usually from author affiliations, sources of funding, and supply and demand relationships.

Some sources of risk of bias are not present here due to the adoption of stringent search, exclusion and inclusion criteria. For example, a frequent source of risk of reporting biases is multiple publication. However, in the present case, this risk was mitigated by the criterion of excluding subsumed publications. Moreover, language biases are not present here due to the adoption of a search string written in the English language and the exclusion of papers written in other languages.

Despite the risk mitigation procedures adopted by the respective authors, included studies may still present residual biases. That is why a risk of bias assessment was developed for each included paper. These papers were assessed individually, covering each of the three sources of risk mentioned above, namely research, reporting and other sources of bias. The checklists of investigated risks appear in catalogofbias.org/biases/ (accessed on 6 December 2021) and Chapter 7 of [111]. The probabilities of occurrence of individual risks are graded as low, unclear and high depending on notable concerns regarding biases. In individual assessments, the baseline risk level is considered to be low. Depending on any notable concern about biases, the perceived risk is increased in one or two levels. Table A1 in Appendix B provides the overall assessment of the risk of bias in each included paper. Therein, each paper's overall risk of bias is computed as the median of the three individual risk grades. Table A2 in Appendix B presents detailed judgments of the identified risks with quotes extracted from the studied papers or texts produced by the author. Finally, Figure 5 summarizes the risk of bias assessment through a diagram generated using the `robvis` tool (www.riskofbias.info/welcome/robvis-visualization-tool, accessed on 6 December 2021) [112].

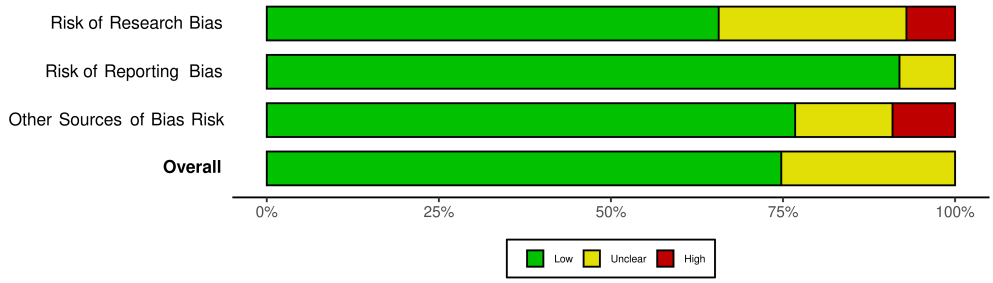

Figure 5. Assessment of Overall Risk of Bias in Included Studies.

5.2. Evaluation of Certainty in the Body of Evidence

According to GRADE, the evaluation of certainty in a body of evidence is based on the risk of bias in each included paper and on the certainty perceived in the respectively reported findings. At the present stage, all included articles are considered in this evaluation, without exclusions due to perceptions of moderate or high risk, since the findings reported in each included paper are useful for triangulation purposes [29].

The level of certainty in each included paper is evaluated as very low, low, moderate or high. The baseline level of certainty in each included article is moderate or low, depending on the reported study type. Reviews, surveys and experiments, and papers that mix many study types, are regarded to have a moderate baseline level of certainty. Articles with studies of other types are regarded to have low baseline certainty. According to some qualifiers, the initial level of certainty may be increased in one level. The level of certainty is increased if a paper reports indirect studies performed in a systematic manner (that is, it corresponds to systematic literature review, systematic mapping or meta-analysis). Other papers have their initial level of certainty upgraded in one level if all the reported studies are performed using randomization or adopting control groups or methods.

The final evaluation of the level of certainty in each included paper is reached by weighing the computed level of certainty, taking into account the perceived risk of bias in the paper. For example, an article with a high calculated level of certainty and low risk of bias is considered a high certainty level. On the other hand, if a paper has a moderate calculated level of certainty and an unclear or high risk of bias, the level of certainty in the paper is downgraded, respectively, to low or very low. The table in Appendix C details the level of certainty computed for each included article.

For many reasons, the certainty gradation used here is different from those adopted in evaluating certainty in other subject areas, such as in Medicine [113]. In other fields, government organizations massively fund scientific research. Consequently, surveys achieve more extensive coverages and experiments present more dramatic effects [110]. In SE, on the other hand, these studies are funded mainly by private organizations or using student and research grants, consequently achieving more modest coverages and more restricted results. Furthermore, in SE, empirical studies such as case studies and simulation studies are important for practitioners, due to budgetary reasons, and they help in investigating hypotheses in specific environments, such as in-house software production processes [103]. As a general rule, in SE, the available objective evidence is comparatively weaker than in other fields and consequently the certainty in individual papers should be evaluated taking this context into account.

5.3. Evidence Profile and Summary of Findings Table

Now, an evidence profile and summary of findings table for the whole systematic mapping is developed. An evidence profile is produced to ascertain the quality in the studied body of evidence. This profile records justified perceptions of aggregated study findings quality based on their risk of bias, review limitations, inconsistency, indirectness and imprecision. Quality is graded here in the way described in Section 5.2. The summary of findings table synthesizes the aggregated evidence.

It is crucial to mention that the specific characteristics of the present study lead to customizations in the GRADE instruments, something allowed by the respective guidelines. Indeed, the GRADE handbook recognizes that the importance of outcomes can vary within and across cultures or when considered from different perspectives [16]. In turn, the literature identifies that SE systematic mappings and literature reviews are mostly qualitative [114]. This happens because, e.g., it is very difficult to observe the same SE phenomena on different occasions. The hypotheses and measures formulated in studies are rarely shared and their results are not easily replicable [24]. Moreover, studies with observational or simulation nature are often useful in SE. So, information required by GRADE sometimes does not exist, such as the number of subjects and the significance levels, confidence intervals and error rates adopted in studies.

Consequently, a particular process of aggregating evidence from included papers is performed. This process considers the productivity factors analyzed in Section 3.4, possibly complemented with evidence supplied by systematic indirect studies. Given that the existence, direction and significance of their relationships have been mapped, a particular process of determining the overall quality of aggregated evidence is required. This process takes the following criteria into account:

1. Remotion of individual studies with low certainty: Only studies with moderate or high certainty are considered;
2. Inclusion of findings that have been deemed collectively important: Only outcomes determined in at least three high- or moderate-certainty papers are considered;
3. Formulation of each aggregated finding definition: Analysis of individual definitions and formulation of an aggregated relationship, involving the productivity factors mentioned in the original studies, any directionality of effects and significance of results, considering the lowest significance and more general scope conclusively reported;
4. Computation of the numbers of papers and studies leading to the finding;

5. Evaluation of the pooled risk of bias for the finding: Computed by weighing the individual paper risk of bias ratings according to the respective number of reported studies and their assessments of risk of bias, using the same criteria of Section 6.1;
6. Determination of inconsistency, indirectness, imprecision and review limitations related to the finding: Usage of the GRADE criteria for determining these aspects;
7. Computation of the overall quality of the finding: Usage of the lowest quality of evidence level among the respective studies as the baseline quality of the outcome, possibly downgraded (according to what was determined in the previous two steps) or upgraded (depending on the findings reported in any systematic indirect study with the same coverage) in one or two certainty levels;
8. Registration of any relevant comment.

Table 17 presents the derived evidence profile and summary of the systematic mapping findings. In the table, there are two distinct categories of study findings. The first one addresses organizational and managerial aspects related to software productivity. The second one is concerned with technical aspects. There is great diversity in the study findings and the quality of evidence varies from high (less frequent) to low (more frequent), pointing out that more focused and authoritative practice-oriented industrial-scale studies regarding software productivity are required.

The findings related to organizational and managerial aspects are confirmatory. The quality of evidence concerning the negative influence of project size on software development productivity is high. It is moderate in the case of the significant negative influence of team size on software project productivity. Professional experience, technical and managerial capabilities are determined to have significant positive impacts on software project productivity with moderate and low levels of certainty, respectively.

The findings related to technical aspects are somehow more diverse. They cover the positive contribution of adoptiing development tools, reuse and rapid application development to software productivity, although these results are not determined with standard significance levels and with high quality of evidence. On the other hand, the influence of programming languages and software artifact complexity on software productivity are obtained with the standard significance levels and moderate quality of evidence. Finally, a negative result concerning the productivity of test-driven development is also determined with the same quality of evidence. Interestingly, the adoption of development tools, modern programming languages, intense software reuse, prototyping and Test-Driven Development are characteristics of lean and agile development methods, which still need to confirm as a whole their positive and significant contributions to software productivity. Indeed, the relationship between productivity and culture in agile methods merits a thorough future investigation [73].

5.4. Methodological Lessons Learned and Recommendations

The methodological lessons reported in the analyzed papers or learned in the process of conducting the present study are now used in the derivation of recommendations for research and practice. These recommendations are derived here from applying the methodology described in the present study or from the contents of the included papers while investigating the systematic mapping research questions. Consequently, they are not a result of any systematic investigation process. Nevertheless, each recommendation is formulated because it was derived in the conduct of present study or it was suggested/implied in/by at least three included papers.

Table 17. GRADE Evidence Profile and Summary of Findings.

Finding	Overall RoB †	Limitations	Inconsistency	Indirectness	Imprecision	# Studies	Papers	Quality	Comments
development project productivity ∼ maintenance project productivity	Low	—	[39]: <<; [50]: ∼∼; [56]: >;	—	—	3	[39], [50], [56]	LOW	Downgraded due to inconsistency
project size ↓→↑ development project productivity	Low	—	—	—	—	6	[57], [86], [108]	HIGH	[59,63]: Similar findings concerning maintenance project productivity with directionality of effect in the opposite direction
team size ↓→→↑ software project productivity	Low	—	—	—	—	4	[39], [43], [53], [108]	MODERATE	—
professional experience ↑→→↑ software project productivity	Unclear	[59]: RoBs come from CoIs ‡	—	—	—	4	[38], [59], [109]	MODERATE	[38]: Reports on experience heterogeneity
technical and managerial capabilities ↑→→↑ software project productivity	Unclear	[23]: RoBs come from risks of research bias	—	—	small samples, missing data, measurement issues	15	[6], [23], [37] *, [96] *, [102]	LOW	Downgraded due to imprecision; [6]: Reports not significant findings
adoption of development tools ↑→↑ development project productivity	Unclear	[23]: RoBs come from risks of research bias	—	—	small samples, missing data, measurement issues	15	[23], [37], [47] **, [56], [77], [108] *	LOW	Downgraded due to imprecision
adopted programming language →→ software project productivity	Low	—	—	—	—	3	[56], [108], [109]	MODERATE	[55,57]: Similar findings concerning development platforms
artifact complexity ↓→→↑ development project productivity	Low	—	—	—	—	4	[37], [86], [101]	MODERATE	[101]: Neither conclusive nor significant findings
software reuse ↑→↑ development project productivity	Low	—	—	—	—	3	[22] *, [80], [81]	MODERATE	[18]: Additional inconclusive evidence
RAD ↑→↑ development project productivity	Low	—	—	—	Direct versus indirect study findings	4	[21], [38] *	LOW	Downgraded due to imprecision; [26,30]: Additional inconclusive evidence
TDD productivity < TLD ♭ productivity	Unclear	[23]: RoBs come from risks of research bias	—	—	small samples, missing data, measurement issues	10	[17], [23], [25]	MODERATE	Downgraded due to imprecision; Upgraded due to the certainty supported by SLRs

† RoB: risk of bias; ‡ CoIs: conflicts of interest; ♭ Test-Last Development; * $\alpha = 0.05$; ** $\alpha = 0.01$.

The importance of deriving methodological recommendations for the research and practice of software productivity is recognized in the literature. Petersen [29] mentions that different approaches should be compared with each other to provide valuable recommendations. Murphy-Hill et al. [89] point out that the impact of productivity research in SE would be improved with a multidimensional toolbox of productivity metrics and instruments, validated through empirical study and triangulation. Despite their generic formulation, these arguments highlight the importance of methodological recommendations as general principles to be considered in software productivity research and practice. In the broader context of SE, Brereton et al. [114] admit that some modifications to standard practices could significantly improve their value as a research tool and a source of evidence for practitioners.

5.4.1. Software Productivity Standards

Standards have paramount importance in ensuring non-ambiguous and uniform understandings of the terms and definitions adopted in the software productivity field. According to Boehm [21], it is vital to establish measurement standards. Indeed, standards related to software productivity provide lists of measures that serve as guidelines for collecting productivity data in different phases of development processes [11].

Two international standards excplicitly address the software productivity theme, ISO 9126-4 and IEEE Std. 1045, but, unfortunately, their adoption in the research and practice communities is not widespread. Still, the literature recognizes the necessity of specific standards. For example, Maxwell, Wassenhove and Dutta [108] identify the need for an international standard for lines of code encompassing all procedural languages. Moreover, Trendowicz and Münch [105], in connection to software productivity standardization, mention as a drawback that many organizations assume measuring software productivity is similar to measuring other forms of productivity. In addition, Cheikhi, Al-Qutaish and Idri [11] suggest that standards would bring convergence and consensus on productivity measures and their factors, facilitating benchmarks and the repeatability and reproducibility of software productivity studies.

> **Lesson 1.** The software productivity community should seek to reduce the uncertainty concerning definitions related to software productivity by participating in standardization initiatives and standardization boards, apart from effectively adopting standards in research and practice.

5.4.2. Practitioner/Industry Involvement and Participation

There are challenges in achieving effective practitioner and industry participation in software productivity studies. Indeed, Bibi, Ampatzoglou and Stamelos [98] recognize that it is difficult to find volunteer professionals for experiments in industrial settings. On the other hand, Kitchenham and Mendes [22] mention the invaluable participation of executives in studies since they may have different perspectives on particular research problems, e.g., by accessing productivity as a more complex attribute than researchers.

It is evident that the potential benefits of collaboration must be made clear for attracting industry practitioners and researchers. Rubin [78] proposes the implementation of corporate dashboards, presenting the selected measures from technological, business, customer and enterprise shareholder perspectives. Lavazza, Liu and Meli [93], in connection with the function point metric, mention that many public administrations and private organizations adopt contractual cost models based on the size of the software to be delivered as the only independent variable. They suggest that empirical studies can help apply the best practices based on objective knowledge, thus avoiding macroscopic mistakes. Based on comparative performance metrics, Tsunoda et al. [53] suggest that project managers should take into account the balance of delivery date and cost in planning their activities, consider a decrease in productivity in replacement projects, the trade-off between loss of productivity and savings in terms of staff costs through the use of outsourcing. These cases illustrate the benefits of participation.

However, there are risks for industry participation in studies, such as disclosing industrial or commercial secrets, and sensitive data or strategies. Software productivity researchers should propose appropriate risk mitigators, such as data anonymization.

> **Lesson 2.** In order to motivate involvement, software productivity researchers should seek the participation of industry practitioners and researchers in studies by presenting them the potential benefits together with the identified risk mitigators.

5.4.3. Software Productivity Data Collection

According to Scacchi [103], to understand the variables that affect software productivity, one has to answer the questions who and what to measure, as well as how to measure productivity. Section 3.2 focused on answering these questions.

Siok and Tian [72] provide some guidelines for data collection in empirical studies on software productivity: make sure that collected data is verifiable and complete, and understand the macro- and micro-level software processes and their assumptions. However, it is sometimes difficult to follow these guidelines. For example, Petersen [29] points out that studies should become more consistent in the way of describing the context and strive for high coverage of context elements.

It is important to take into account not only the convenience for stakeholders but also the employed resources in data collection. Indeed, Scacchi [103] identifies that programmer and manager self-reported data are the least costly to collect, although they may be of limited accuracy, and that outside observers can often collect such information but at a higher cost than self-report. Moreover, automated tools are recognized to be useful, but require more insight into what should be measured and how [103].

Another concern is the scope of data collection processes. Kitchernham and Mendes [22] mention the definition of random samples from well-defined populations as an outstanding problem. They also identify the need for methods for drawing conclusions from nonrandom and quasi-random datasets.

> **Lesson 3.** Software productivity data analysts should be concerned with data collection processes and data quality. They should always characterize the context and population under study in a precise way; propose in a justified manner sample, experiment or case study size; describe data sources, studied variables and data collection processes, with their time spans and collection instruments. Whenever possible, randomization should be adopted.

5.4.4. Usage of Productivity and Open-Source Code Databases

The efforts to standardize data definition and collection in public databases and OSS code repositories have been considered relevant and welcome, as mentioned in Section 3.2. According to Maxwell, Wassenhove and Dutta [108], data validity and comparability is maximized as all companies collect data using the same tool and every variable is defined. Lavazza, Liu and Meli [56] also point out that many public databases projects represent consolidated practices and languages.

However, there are many challenges in adopting databases and repositories for software productivity research. Although there have been some national and international research organizations responsible for creating specific projects for establishing databases of productivity measures along with the respective factors, in general, these projects have ended fading away, as noted by Hernández-López et al. [32]. Indeed, according to Premraj et al. [50], among the challenges that researchers face there are the potential risks with analyzing complex datasets without good channel communication with those associated with the actual dataset collection. Moreover, there is suspicion that innovative applications will always be in the minority in these source, given their recentness [56]. In addition, according to Rodríguez-García et al. [39], extensive reprocessing is required to apply statistical or data mining techniques on public databases due to ambiguities, missing values, unbalanced datasets, etc.

Balancing the challenges and opportunities of public database and repository adoption in software productivity research, there is good potential for practical and useful applications. For example, it would be possible to use them as registers serving as sources of information concerning studies being carried out (before they are published). This practice would facilitate the assessment of risk of bias in studies [111].

> **Lesson 4.** Software productivity data scientists should seek to adopt and expand the practice of compiling productivity databases towards exploring new and innovative applications, taking into account the best practices and the associated challenges and opportunities.

5.4.5. Software Productivity Measurement and Analysis

As discussed in Section 3.2, it is a good practice to choose data analysis methods that best fit the problem under analysis, but the preconditions for their application are frequently not discussed in published papers on software productivity.

The first and foremost requisite for software productivity analysis is to understand productivity measurement. There may be dimensionality in this activity and the definition of the respective scales should be an initial concern . According to Scacchi [103], the efforts to develop productivity measures for large-scale systems may lead one away from traditional quantitative measures towards symbolic and qualitative models that incorporate nominal, ordinal, interval and ratio measures. In addition, Cheikhi, Al-Qutaish and Idri [11] argue that productivity measures may be multidimensional and consider quality factors. Furthermore, Storey et al. [47] point out that productivity is multi-faceted (i.e., various factors influence it) and highly perceptual, since capturing developers' views of their own productivity can be a way to measure performance.

According to Hernándes-Lópes et al. [32], the analysis level should be considered in defining productivity measures, as they may cover a country, a business sector, an organization, a department, a project, a unit or an individual. Diverse measurement goals may exist in different contextual levels. This classification can be used not only for productivity factors, but also for software productivity itself [105].

Boehm [21] highlights that there are two primary ways of analyzing software productivity: the "black-box" or influence-function approach, and the "glass-box" or cost-distribution approach. Both make sense from a corporate perspective and managers should choose the approach that better serve their needs. Siok and Tian [72] provide guidelines for software productivity measurement and analysis: understand how the applicable analysis methods work and when to use them, and make corporate decisions based on the data and their analysis.

The comprehension and discussion of adopted analysis methods should be performed to justify any choice and increase confidence in study findings. The best practices should be considered. In case statistical methods are adopted, the discussion of frequency distribution, missing data, homoscedasticity, colinearity, goodness of fit, statistical power and effect size should be done whenever appropriate.

> **Lesson 5.** Software productivity data analysts should choose productivity measurement and analysis methods considering the problem at hand. They should take into account the measurement level and approach, the corporate goals and the best practices in terms of analysis methods.

5.4.6. Confounding Factors

As identified in Section 6.2, few papers discuss confounding factors related to software productivity. Moreover, Petersen [29] points out that it is not always clear whether or not there are unknown confounding factors that influence study outcomes.

A first step to treat this situation consists in clarifying the distinction between productivity and other SE dimensions in each study. For example, Kitchenham and Mendes [22] mention the possible confounding between productivity and size differences. Cardoso et al. [26] point out that outcomes related to project performance may be confounded to

productivity, such as customer satisfaction, product and process quality, team motivation, and cost reduction.

In addition, it is vital to identify confounders using specific knowledge regarding the studied context and problem. Management and process changes and product maturity can be considered confounding software productivity factors in a corporate environment [9]. However, process and product maturity are not entirely orthogonal and may also be confounded [40]. Many other variables may correspond to confounding factors. Tsunoda and Amasaki [55] list development type, unadjusted FP, duration, business sector, development platform, programming language and FP attributes as potential confounders. Mohagheghi and Conradi [18] mention context, size, programming language, complexity, task and methods concurrency, skills and knowledge. Bibi, Ampatzoglou and Stamelos [98] regard upgrading a system version to another one as a confounding factor. As shown, the same variables can be regarded as productivity factors or as confounders depending on the study setting.

It is interesting to note that studies adopting contemporary data collection and analysis techniques or addressing emerging topics in SE introduce novel potential confounding factors in studies. For example, Krein et al. [94] considers that months without submissions to software repositories represent a confounding factor in developer productivity studies. Rafique and Misic [17] mention that the simultaneous adoption of other agile practices might confound the effects of TDD. Mantyla et al. [60] identify that, as a consequence of removing or disguising emotions in developer communications, comments collected in chats may become a cause of confusion. In connection to self-assessed productivity, Kuutila et al. [102] point out that experiences and events not related to work can have a confounding effect on mixed-effects models.

> **Lesson 6.** Authors of software productivity studies should clarify and analyze the software engineering dimensions that may be confounded with software productivity and the factors that may confound software productivity analysis.

5.4.7. Conduction of Studies on Software Productivity

Finally, the lessons learned in conducting the present study are presented. They are related to the formulation of research questions, the adoption of taxonomies, the writing of search strings and the avoidance of conflicts of interest in studies, particularly in literature reviews and systematic mappings.

Regarding formulating research questions in empirical studies on software productivity, at least two different approaches have been adopted. Here, the Goal Question Metric (GCM) methodology [19] is used since it was defined within the context of SE and appeared to be more aligned to the present study's specific subject and general objectives. A similar approach was adopted in [18]. In other SE studies, such as in [25,27,29], and different fields, such as Medicine, the Patient Intervention Comparison and Outcome (PICO) approach has been preferred. Both approaches help formulate research questions and facilitate the search for precise answers.

Although the GRADE system [16] explicitly prescribes PICO adoption, some difficulties in framing software productivity problems according to this approach exist. In particular, studies with observation, analysis and description goals would not be entirely compatible with the requirement of formulating intervention and outcome elements. In turn, research questions for measurement and prediction studies would have to be formulated in specific ways to comply with the PICO framework. From the methodological perspective, strictu sensu, interventions and outcomes would only be addressed in studies with action goals.

> **Lesson 7.** Authors of software productivity studies should prefer GCM over PICO. The adoption of PICO should always be justified in terms of the study goals and characteristics.

Another aspect that deserves attention is the use of SE taxonomies in software productivity studies. Even though adopting the SWEBOK KAs [33] here was the basis for study and paper classification, identifying specific KAs in papers was very time-consuming. This situation mainly happened because contemporary SE subjects are marginally addressed in the SWEBOK, such as those elicited in Sections 3.3 and 3.4: agile/lean practices, web development techniques, service-oriented architectures (e.g., SaaS), global development and others. For facilitating study and paper classification, it would be paramount to update the SWEBOK contents and provide therein more practice-oriented guidance, which would facilitate practitioner adoption.

> **Lesson 8.** The IEEE SWEBOK should be updated to cover emergent software engineering subjects and should contain more practice-oriented guidance.

A word of caution is required regarding the formulation of search strings for systematic mappings and literature reviews. While adopting multiple alternative search keys may return a nearly intractable number of references, extremely narrow search criteria (or even mistakes in formulating queries) may miss important references that should be analyzed. In retrospect, it is also possible that a search string that produced a reasonable result on one occasion will fail to have the same results in the replication of a study, as reported in Section 2.4. Consequently, search strategies should be formulated considering variations in the search string and the adopted bibliographic reference databases, apart from adopting alternative methods of reference discovery.

> **Lesson 9.** Authors of systematic literature reviews and mappings on software productivity should formulate strategies of paper screening considering variation in the adopted search string and bibliographic reference databases, apart from using alternative methods of reference discovery.

It is also important to highlight the importance of evaluating conflicts of interest for determining the findings of the present study, as they may have impacted included study design, conduct and reporting [111]. Conflicts of interest were identified as the second most frequent cause of perceived risks of bias in included papers, before risks of research biases and ahead of risks of publication bias, as reported in Section 6.1. The most frequent justifications for these perceptions of conflict were author affiliations and sources of data, technology or funding. Despite this, it is crucial to recognize that perceived conflict identification was possible only due to the reporting transparency of the included papers.

In order to manage or avoid conflicting situations, it would be necessary for authors of empirical studies on software productivity to adopt specific guidelines for ensuring research quality and transparent reporting. These comprise clear and explicit statements of author affiliations, sources not only of funding but also of technology and data, as well as of conflicts of interests in papers. The incentives for study participation and disclosure limitations on research data and findings should also be reported.

> **Lesson 10.** Authors of software productivity studies should ensure research quality and transparent reporting by including in their papers clear and explicit statements of author affiliations, sources of funding, technology and data, and conflicts of interests, apart from transparently reporting incentives for study participation and disclosure limitations on research data and findings.

These learned lessons and recommendations for industrial practice are not complete and should be used in conjunction with others formulated from different perspectives (cf. [110,114]).

6. Threats to Validity

6.1. Construct and Internal Validity

Systematic mappings and literature reviews are susceptible to subjectiveness and inaccuracy in the chosen notions and the lack of rigor and precision in the formulated definitions, leading to construct validity threats. In order to mitigate the former kind of

threat, the adopted notions were discussed with experienced researchers in meetings and conferences. The latter type of threat was mitigated by selecting authoritative taxonomies whenever available. That is why the definitions in the SWEBOK [33] and the study types and productivity approaches defined in [18] were adopted here.

The extent to which the design and conduct of each systematic mapping and literature review are likely to prevent systematic error, that is, their internal validity, is threatened by reviewer biases. Here, the main threat to internal validity is that a single researcher conducted the entire study. In order to mitigate the associated threats, the procedures and findings reported here were manually checked and later rechecked using the ROBIS tool (www.robis-tool.info, accessed on 24 April 2022) [115]. First, the tool identifies concerns with the review process by assessing study eligibility criteria, study identification and selection procedures, data collection and study appraisal, and synthesis and findings. Next, a judgment is reached concerning the possibility of review bias. The self-application of the tool questionnaire in the present case resulted in a judgment of low risk of bias. The dissemination of the review data and protocol in the way described in the Supplementary Materials item ensures additional confidence and transparency of the reported findings.

The possibility of bias in including publications for review also threatens internal validity. The standard way to avoid paper selection threats is to follow a definite research methodology and review protocol. In the present study, the recommendations suggested by Kitchenham and Charters in [14] and the PRISMA guidelines [15] were simultaneously adopted, apart from a pre-established review protocol. However, the choice of the year 1987 to define the beginning of the reference search period could represent a threat to the reported research. Although this choice was arbitrary, considering the existence of a series of impactful publications from that year onward, previous publications most likely would not comply with exclusion and inclusion criteria, if they were available online for analysis. Moreover, including the few remaining publications in the present study would not significantly affect the systematic mapping findings.

An additional source of internal validity threats in systematic mappings and literature reviews is the existence of relevant undetected papers. Indeed, [14] alerts that no single search can find all relevant studies. In the present case, DBLP and Scopus were adopted as sources of bibliographic references. DBLP is an open and curated tool covering the most relevant sources of SE research and Scopus is one of the main expertly curated sources of scientific research. Recursive backward snowballing was also adopted to identify additional bibliographic references. The paper screening process returned 495 references, from which 99 papers were selected for inclusion in this study. The number of included papers is more extensive than those mentioned in each line of Table 16. Despite the mitigators, it is essential to recognize that many other studies on software productivity based on empirical methods exist, particularly those published in the proceedings of regional events and journals. Still, in practice, it is almost impossible to cover all regional sources of publication concerning SE while ensuring fairness of treatment, due to constraints such as paper availability and knowledge of many different foreign languages. For the same reason, apart from the fact that it is difficult to identify in SE [29], the Grey literature was not reviewed here. Furthermore, it is important to point out the lack of online paper availability as another source of similar threats. However, this is a usual limitation in systematic mappings and literature reviews, as recognized in most of the related work analyzed in Table 16.

The validity of included studies may be threatened by confounding factors, which make it impossible to distinguish the effects of two interventions from each other. The SWEBOK [33] lists economic friction (everything keeping markets from having perfect competition), ecosystems and outsourcing/offshoring as confounding factors related to software engineering economics in general and software productivity in particular. However, confounding depends on the context of each study and require specific knowledge to be identified. Still, confounding factors are rarely studied in the reviewed literature [18], providing evidence that the risk of bias due to confounding is regarded as low in included

papers. This situation suggests that additional attention is needed to confounding factors in studies related to software productivity.

6.2. External Validity

External validity corresponds to the extent to which the reported results are reliable and can be generalized to other populations and settings [14]. It is challenging to perform external valitidy analyses of concerning literature reviews and systematic mappings because they analyze and synthesize the diverse findings of other studies. Nevertheless, the analyzed papers correspond to a representative sample of the industrial practice of software productivity in the studied period and the adopted methodology is sufficiently transparent to be replicated considering other time frames and studies.

7. Concluding Remarks

This paper provides evidence of different empirical perceptions of software productivity within the distinct business sectors and KAs covered in the industrial practice of SE. There are also many commonalities in approaching software productivity in these KAs and sectors, primarily due to the adopted analysis methods and their respective measures. The research findings in included studies were analyzed and synthesized and a list of recommendations for industrial research and practice was derived based on lessons learned in the respective papers and in conducting the present study.

The main contributions of the reported research have practical and methodological significance. From the methodological perspective, applying the PRISMA guidelines in SE, as outlined here, is innovative and demonstrates the feasibility of borrowing empirical study analysis methods from other fields, particularly the GRADE system from the healthcare sector. In addition, it is expected that the set of methodological recommendations derived here will help industry practitioners in addressing the software productivity subject and developing further research. From the practical perspective, factors that affect software productivity were elicited from included studies and classified according to organizational/managerial and technical categories. In particular, the reported research demonstrates that the impacts of agile development practices on software productivity have great variability and still need to confirm their positive and significant contributions.

The strengths/trends and weaknesses/gaps in analyzed studies suggest directions for future research. A more holistic approach in software productivity studies is needed [11], covering more or unabridged KAs (SR and SEMM), sectors (noticeably industry, retail and health care) and environmental factors (economic friction and ecosystems), while treating the lack of standardization and sufficient reporting in studies. The development of more confirmatory, replication [50] and multi-company studies is required [32], together with studies with analysis, description and action goals. Over the years, the historical evolution of this field—with a noticeable increase in the number of published studies—provides continued evidence that software productivity is still considered an important subject within SE. Only with more authoritative practice-oriented industrial-scale studies will the quality and certainty in the body of evidence increase.

Supplementary Materials: The data extraction protocol and tabular data obtained in the paper screening process are publicly available on www.chcduarte.com/ReviewedLitetratureOnSoftwareProductivity2021.xlsx (accessed on 24 April 2022) at doi:10.13140/RG.2.2.26436.35205/1.

Funding: This research received no external funding.

Conflicts of Interest: The assumptions, views and opinions in this article are solely those of the author and do not necessarily reflect the official policy, strategy or position of any Brazilian government entity. The author declares no conflict of interest.

Appendix A. Coding of Factors Affecting Software Productivity

Appendix A.1. Studies Using SE Economics Databases

1. architecture →→ development project productivity ([53,56]);
2. development platform →→ development project productivity ([55]);
3. business sector □ software project productivity
   (where □ =→→ for [56]; □ =→ for [49,52,53]);
4. team size ↓ □ ↑ development project productivity
   (where □ =→→ for [53]; □ =→? for [52]);
5. adopted programming language □ development project productivity
   (where □ =→→ for [56]; □ =→? for [52]);
6. company → development project productivity ([49,50]);
7. project size ↑→↑ development project productivity ([52]);
8. level of outsourcing ↓→↑ development project productivity ([53]);
9. adopted programming language → maintenance project productivity ([56]);
10. adoption of development tools →? development project productivity ([56]);
11. development project productivity □ maintenance project productivity
    (where □ =<< for [39]; □ => for [56]; □ =∼∼ for [50,52]);
12. large team development productivity < small team development productivity ([39]).

Appendix A.2. Other Studies Covering the SWEBOK

1. formal education ↑→→↑ labor productivity ([106]);
2. organizational structure →→ development project productivity ([108]);
3. risk classification →→ development project productivity ([57]);
4. UCPs →→ development project productivity; ([86]);
5. FPs →→ development project productivity ([57]);
6. LOCs ↑→→↑ development project productivity ([108]);
7. development platform →→ development project productivity ([57]);
8. software complexity ↓→→↑ development project productivity ([86]);
9. adopted programming language recency ↑→→↑ development project productivity ([109]);
10. team experience ↑→→↑ development project productivity ([109]);
11. experience with user community ↑→→↑ development project productivity ([109]);
12. team size ↓→→↑ development project productivity ([108]);
13. application type → development project productivity ([108]);
14. software reuse ↑→↑ development project productivity ([61]);
15. technical debt ↓→↑ development project productivity ([87]);
16. software development approach adequacy ↑→↑ scientific software productivity ([83]);
17. RAD ↑→↑ development project productivity ([21]);
18. adoption of development tools → development project productivity ([61,108]).
19. adopted programming language → development project productivity ([108]);

Appendix A.3. Requirements Engineering

1. requirements volatility ↓→↑ development project productivity ([9]);
2. requirements engineer communication ↑→↑ development project productivity ([9]);
3. adoption of requirement management tools → development project productivity ([66]).

Appendix A.4. Object-Oriented Development

1. project size ↑→→↑ development project productivity ([48]);
2. application domain → development project productivity ([72]);
3. adoption of OOD → development project productivity ([69]);
4. rigorous enforcement of project deadlines ↓→↑ development project productivity ([68]);
5. early intermediate task completion incentives ↑→↑ development project productivity ([68]).

Appendix A.5. Software Construction

1. software reuse ↑ □ ↑ development project productivity
   (where □ =→→ for [22]; □ =→ for [71]);
2. formal education ↑→↑ development project productivity ([23]);
3. architecture → development project productivity ([84]);
4. requirements volatility ↓→↑ development project productivity ([71]);
5. knowledge of unit testing ↑→↑ development project productivity ([23]);
6. team capabilities ↑→↑ development project productivity ([71,84]);
7. adoption of development tools ↑→↑ development project productivity ([23,71]);
8. concurrent development pair productivity < simultaneous development pair productivity ([97]).

Appendix A.6. Software Reuse

1. software reuse ↑ □ ↑ development project productivity
   (where □ =→→ for [62]; □ =→ for [64,80]; □ =→? for [81]).

Appendix A.7. Open-Source Software

1. adopted programming language fragmentation ↓→→→↑ OSS project productivity ([94]);
2. OSS adoption ↑→→↑ service corporate labor productivity ([100]);
3. OSS age ↓→→↑ OSS project productivity ([95]);
4. team size ↓→→↑ OSS project productivity ([43]);
5. team experience ↑→↑ OSS project productivity ([36]);
6. LOC-based size increment ↑→↑ OSS project project productivity ([41]).

Appendix A.8. Software Testing

1. project difficulty ↓→↑ testing project productivity ([67]);
2. skilled programmer task process transference ↑→↑ testing project productivity ([75]).

Appendix A.9. Software Maintenance

1. domain knowledge ↑→→↑ maintenance project productivity ([96]);
2. team capabilities ↑→→↑ maintenance project productivity ([96]);
3. mentors succession and experience ↑→→↑ maintenance project productivity ([59]);
4. mentors work load ↓→→↑ maintenance project productivity ([59]);
5. level of offshoring succession ↓→→↑ maintenance project productivity. ([59]);
6. project size ↓ □ ↑ maintenance project productivity
   (where □ =→→ for [59]; □ =→ for [63]);
7. LOC-based size increment ↑→↑ maintenance project productivity ([63]);
8. maintenance granularity ↓→↑ maintenance project productivity ([98]);
9. software artifact coupling ↓→↑ maintenance project productivity ([98]);
10. project quality ↓→? ↑ maintenance project productivity ([96]).

Appendix A.10. Software Engineering Management

1. project size $\uparrow\rightarrow\rightarrow\rightarrow\uparrow$ development project productivity ([35]);
2. adoption of development tools $\uparrow\rightarrow\rightarrow\rightarrow\uparrow$ development project productivity ([47]);
3. adoption of process models $\uparrow\rightarrow\rightarrow\rightarrow\uparrow$ development project productivity ([47]);
4. team autonomy $\uparrow\rightarrow\rightarrow\rightarrow\uparrow$ development project productivity ([47]);
5. technology knowledge $\uparrow\rightarrow\rightarrow\uparrow$ development project productivity ([37]);
6. RAD $\uparrow\rightarrow\rightarrow\uparrow$ development project productivity ([35,38]);
7. team experience heterogeneity $\downarrow\rightarrow\rightarrow\uparrow$ development project productivity ([38]);
8. adoption of testing tools $\uparrow\rightarrow\rightarrow\uparrow$ development project productivity ([37]);
9. task coordination $\uparrow\rightarrow\rightarrow\uparrow$ development project productivity ([40]);
10. software complexity $\downarrow\rightarrow\rightarrow\uparrow$ development project productivity ([37]);
11. task completion incentives $\downarrow\rightarrow\rightarrow\uparrow$ development project productivity ([47]);
12. possibility of mobility $\downarrow\rightarrow\rightarrow\uparrow$ development project productivity ([47]);
13. in-house development project productivity $\square$ offshored development project productivity (where $\square$ $=<<$ for [38], $\square$ $=>$ for [42]).

Appendix A.11. Rapid Application Development

1. team management $\rightarrow$ agile software development productivity ([6,12]);
2. team size $\rightarrow$ agile software development productivity ([12]);
3. team diversity $\rightarrow$ agile software development productivity ([12]);
4. team turnover $\rightarrow$ agile software development productivity ([12]);
5. personal capabilities $\rightarrow$ agile software development productivity ([6,12,102]);
6. Scrum adoption $\rightarrow$ software development productivity. ([73]);
7. traditional project productivity $<$ Scrum-RUP project productivity ([58]).

Appendix A.12. Software Processes, Quality, Models and Methods

1. organizational structure $\rightarrow$ development project productivity ([78]);
2. personal software process maturity levels $\rightarrow$ software developer productivity ([82]);
3. proof size $\rightarrow$ formal verification productivity ([101]);
4. appraised software process maturity levels $\square$ corporate labor productivity (where $\square$ $=\rightarrow$ for [78]; $\square$ $=\rightarrow?$ for [54]);
5. adoption of development tools $\rightarrow?$ development project productivity ([77]);
6. proof complexity $\rightarrow?$ formal verification productivity ([101]).

Appendix B. Risk of Bias Assessment Tables

See in Table A1 the assessment of the overall risk of bias in each included study.

Table A1. Assessment of Risk of Bias in Each Included Study.

Key	Risk of Bias Domains			
	D1	**D2**	**D3**	**D4**
(AbdelHamid96) [79]	Low	Low	Low	Low
(AdamsCB09) [36]	Unclear	Low	Low	Low
(AsmildPK06) [70]	Low	Low	Low	Low
(AzzehN17) [85]	Low	Low	Low	Low
(AzzehN18) [86]	Low	Low	Low	Low
(BankerDK91) [96]	Low	Low	Low	Low
(BankerK91) [62]	Unclear	Low	High	Unclear
(BankerS94) [63]	Low	Low	Low	Low
(BellerOBZ21) [74]	Low	Low	High	Unclear
(BeskerMB19) [87]	High	Low	Low	Unclear
(BezerraEA20) [88]	Unclear	Low	Low	Low
(BibiAS16) [98]	Low	Low	Low	Low
(BibiSA08) [91]	Unclear	Low	Low	Low
(Boehm87) [21]	Low	Low	Low	Low
(Boehm99a) [80]	Low	Low	Low	Low
(CarvalhoRSCB11) [58]	Low	Unclear	Low	Unclear
(CataldoH13) [40]	Low	Low	Unclear	Low
(ChapettaT20) [24]	Low	Low	Low	Low
(Chatman95) [65]	Unclear	Low	High	Unclear
(CheikhiARI12) [11]	Low	Low	Low	Low
(DamianC06) [9]	Low	Low	Low	Low
(DiesteEtAll17) [23]	High	Low	Low	Unclear
(Duarte17a) [54]	Unclear	Low	Low	Low
(Duncan88) [61]	Low	Low	High	Unclear
(FatemaS17) [6]	Low	Low	Low	Low
(FaulkLVSV09) [83]	Low	Unclear	High	Unclear
(FrakesS01) [81]	Low	Low	Low	Low
(GeH11) [90]	Low	Low	Low	Low
(GraziotinWA15) [4]	High	Low	Low	Unclear
(GreenHC05) [82]	Unclear	Low	Low	Low
(HenshawJMB96) [66]	Low	Low	Unclear	Low
(HernandezLopezCSC15) [7]	High	Low	Low	Unclear
(HernandezLopezPGC11) [32]	Low	Low	Low	Low
(HuangW09) [99]	Unclear	Low	Low	Low
(JaloteK21) [75]	Unclear	Low	Unclear	Unclear
(JohnsonZB21) [76]	Unclear	Unclear	High	Unclear
(KautzJU14) [73]	Low	Low	Low	Low
(KemayelMO91) [109]	Low	Low	Low	Low
(KitchenhamM04) [22]	Unclear	Low	Low	Low
(KreinMKDE10) [94]	Low	Low	Low	Low
(KuutilaMCEA21) [102]	Low	Low	Low	Low
(LagerstromWHL12) [57]	Low	Low	Low	Low
(LavazzaMT18) [56]	Unclear	Low	Low	Low
(LavazzaLM20) [93]	Unclear	Low	Low	Low
(LiaoEA21) [95]	Low	Low	Low	Low

Table A1. *Cont.*

Key	Risk of Bias Domains			
	D1	**D2**	**D3**	**D4**
(Lim94) [64]	Low	Low	High	Unclear
(LowJ91) [77]	Low	Low	Low	Low
(MacCormackKCC03) [35]	Unclear	Low	Unclear	Unclear
(MantylaADGO16) [60]	Low	Low	Low	Low
(Maxwe96) [108]	Low	Low	Unclear	Low
(MaxwellF00) [49]	Low	Low	Unclear	Low
(MeloCKC13) [12]	Low	Low	Low	Low
(MeyerBMZF17) [5]	Unclear	Low	Low	Low
(MeyerZF17) [10]	Unclear	Low	Unclear	Unclear
(MinetakiM09) [104]	Low	Low	Low	Low
(MoazeniLCB14) [41]	High	Low	Low	Unclear
(Mockus09) [59]	Low	Low	High	Unclear
(Mohapatra11) [37]	Low	Low	Low	Low
(MosesFPS06) [51]	Unclear	Low	Low	Low
(MurphyHillEA21) [89]	Low	Low	Low	Low
(OliveiraEA20) [46]	Unclear	Low	Low	Low
(PalaciosCSGT14) [42]	Unclear	Low	Low	Low
(ParrishSHH04) [97]	Low	Low	Low	Low
(PortM99) [69]	Low	Low	Unclear	Low
(PotokV97) [68]	Low	Low	High	Unclear
(PotokVR99) [48]	Low	Low	Unclear	Low
(PremrajSKF05) [50]	Low	Low	Unclear	Low
(RamasubbuCBH11) [38]	Low	Low	Unclear	Low
(RastogiT0NC17) [45]	Low	Low	Low	Low
(RodriguezSGH12) [39]	Low	Low	Low	Low
(Rubin93a) [78]	Low	Low	Low	Low
(Scacchi91) [103]	Low	Low	Low	Low
(ScholtesMS16) [43]	Low	Low	Low	Low
(SentasASB05) [92]	Low	Low	Low	Low
(SiokT07) [72]	Low	Low	Unclear	Low
(SovaS96) [67]	Low	Low	Low	Low
(StaplesEA14) [101]	Low	Low	Low	Low
(StoreyEA21) [47]	Unclear	Low	Low	Low
(StylianouA16) [44]	Low	Low	Low	Low
(Tan09) [84]	Low	Low	Low	Low
(TanihanaN13) [100]	Low	Low	Low	Low
(TomaszewskiL06) [71]	Low	Low	Low	Low
(TrendM09) [105]	Low	Low	Low	Low
(Tsuno09) [53]	Low	Low	Low	Low
(TsunodaA17) [55]	Low	Low	Low	Low
(Wang12) [106]	Low	Low	Low	Low
(WangWZ08) [52]	Low	Low	Low	Low
(YilmazOC16) [3]	Low	Low	Low	Low
(ZhaoWW21) [107]	Low	Low	Low	Low
(BissiNE16) [25]	Unclear	Low	Low	Low
(CardozoNBFS10) [26]	Unclear	Unclear	Low	Unclear
(HernandezLopezPG13) [8]	Unclear	Unclear	Low	Unclear
(MohagheghiC07) [18]	High	Low	Low	Unclear
(OliveiraVCC17) [27]	Unclear	Unclear	Low	Unclear
(OliveiraCCV18) [28]	Unclear	Unclear	Low	Unclear
(Peter11) [29]	Unclear	Low	Low	Low
(RafiqueM13) [17]	Low	Low	Low	Low
(ShahPN15) [30]	Unclear	Unclear	Low	Unclear
(WagnerR08) [2]	High	Low	Unclear	Unclear

D1 = risk of research bias, **D2** = risk of reporting bias, **D3** = other Sources of risk of bias, and **D4** = overall risk of bias.

See in Table A2 the justifications for increasing the risks of bias levels perceived in some included studies.

Table A2. Justifications for Increasing the Risk of Bias Levels Perceived in Included Studies.

Key	Explanation for Downgrading
(AdamsCB09) [36]	Bug-tracking data were disregarded and only actual commits studied (observation risk);
(BankerK91) [62]	"Our final sample of 20 projects excluded one project among the initial 21 that was believed to be an outlier" (exclusion risk); "Bedell's alternative strategy to cope with this 'functionality risk' was to build the ICASE tool in house. Although the investment posed a major risk to the firm, First Boston Bank subsequently committed $65 million", "This article addresses three principal research questions: did reusability lead to any significant productivity gains during the first two years of the deployment of the ICASE tool" (conflicting interests risk, studied tool financially supported by the company that demanded the study);
(BellerOBZ21) [74]	"We start to bridge the gap between them with an empirical study of 81 software developers at Microsoft" (conflicting interests risk, due to the authors' affiliation);
(BeskerMB19) [87]	"This study's selection of participating companies was carried out with a representative convenience sample of software professionals from our industrial partners" (selection-availability risk). "On average, each respondent reported their data on 11 out of 14 occasions" (missing data or non-response risk);
(BezerraEA20) [88]	"The survey used two approaches: (i) we used self-recruitment, sharing posts to invite members of social networking groups related to IT professionals on Facebook, Instagram and mailing lists; and, (ii) we sent out direct invitations to people we knew" (selection-availability risk);
(BibiSA08) [91]	"Although there are many missing values in the above fields (over 72%) and the extracted rules have low values of confidence, the results are satisfactory" (missing data risk);
(CarvalhoRSCB11) [58]	14 samples were analyzed, but data were collected regarding 16 projects (selective reporting risk);
(CataldoH13) [40]	"We collected data from a multinational development organization responsible for producing a complex embedded system for the automotive industry" (conflicting interests risk, due to the affiliation of an author);
(Chatman95) [65]	"Current data retention does not preserve all the data implied by the change-point approach, so the results shown in the figures are incomplete" (missing data risk);"The figures present data collected for three releases of a product developed at IBM's Santa Teresa Laboratory" (conflicting interests risk, due to the affiliation of the author);
(DiesteEtAll17) [23]	"The experimental subjects were convenience sampled" (selection-availability risk); "Although we had 126 experimental subjects, 11 observations were lost during the analysis as two subjects failed to complete the experimental task, six failed to report their academic qualifications and four failed to report any experience" (missing data or non-response risk); "Each quasi-experiment was measured by a single measurer" (measurement risk);
(Duarte17a) [54]	"Since our economic data set is sparse, in the sense that there are some missing observations in the middle of some periods, we used interpolation" (missing data risk);
(Duncan88) [61]	"The paper describes the software development process used within one software engineering group at Digital Equipment Corporation", "The questions that the Commercial Languages and Tools software product engineering group at DEC asked are: how are we doing compared to ourselves in previous years? Can we quantify the impact of using software development tools?" (conflicting interests risk, due to the affiliation of the author);
(FaulkLVSV09) [83]	"We ran a set of experiments", but only reduction ratio was reported (selective reporting risk); "Sun Microsystems took a broad view of the productivity problem", "We studied the missions, technologies and practices at government-funded institutions", "DARPA programmatic goal was to address 'real productivity"' (conflicting interests risk, due to author's affiliations and the source of funding);
(GraziotinWA15) [4]	"The participants have been obtained using convenience sampling" (selection-availability risk); "When questioned about the difficulties and about what influenced their productivity, the participants found difficulties in answering" (measurement-recall risk);
(GreenHC05) [82]	"A few respondents noted that it was too early to assess productivity gains. Therefore, some respondents did not respond to productivity related items" (non-response risk);
(HenshawJMB96) [66]	"In the organization we studied, requirements planning had been done using the AIX file and operating system" (conflicting interests risk, studied technology supplied by the employer of an author);
(HernandezLopezCSC15) [7]	"One of the authors contacted via e-mail ex-alumni with experience of at least one year in any activities of SE. From these, 15 positive answers were obtained. Interviews were conducted between April and October 2011" (selection-availability risk); "The authors wrote some posts in LinkedIn groups related to SE. 31% of the respondents accessed the questionnaire from LinkedIn" (selection-inception risk);
(HuangW09) [99]	"Since we do not have access to the proportion of SaaS revenue in a software company, we need to subjectively decide whether its SaaS operations are significant enough so that the target firm is coded as a mixed-SaaS firm. The other source of data limitations is that some firms do not mention their SaaS business in the annual report, or use a different name for SaaS services that is not captured by our Java program" (observation risk);
(JaloteK21) [75]	"As the data were not normally distributed, the Kruskal–Wallis non-parametric test was conducted after removing the outlier" (exclusion risk)" (exclusion risk); "We conducted this field study at Robert Bosch Engineering and Business Solutions Ltd (RBEI)" (conflicting interests risk, due to the affiliation of an author);
(JohnsonZB21) [76]	"We sent the survey to 1,252 individuals with an engineer or program management position at Microsoft in the Puget Sound area" (selection risk); "Design with a total of 1159 participants" and "We sent the survey to 1252 individuals" (selective reporting risk); "To address the lack of empirical data on work environments in software development, we carried out an empirical study of physical work environments at Microsoft" (conflicting interests risk, due to the authors' affiliation);
(LavazzaMT18) [56]	"In the derivation of models, outliers, identified based on Cook's distance, following a consolidated practice, were excluded" (exclusion risk);
(LavazzaLM20) [93]	"Data points with Cook's distance greater than 4/n (n being the cardinality of the training set) were considered for removal" (exclusion risk);
(Lim94) [64]	"The reusable work products were written in Pascal and SPL, the Systems Programming Language for HP 300 computer system", "The development operating system was HPUX" (conflicting interests risk, studied technology supplied by the employer of the author);
(MacCormackKCC03) [35]	"We removed from the analysis projects that were outliers on each performance dimension on a case-by-case analysis" (exclusion risk); "Our results are based on a sample of HP software development projects" (conflicting interests risk, studied technology supplied by the employer of an author);
(Maxwe96) [108]	"We present the results of our analysis of the European Space Agency software development database" (conflicting interests risk, studied projects funded by the research financial supporter).

Table A2. *Cont.*

Key	Explanation for Downgrading
(MaxwellF00) [49]	"The project grew and is now an STTF-managed commercial activity" (conflicting interests risk, studied database supplied by the employer of an author);
(MeyerBMZF17) [5]	"We used personal contacts, e-mails and sometimes a short presentation at the company to recruit participants" (selection-availability risk);
(MeyerZF17) [10]	"We advertised the survey by sending personalized invitation emails to 1600 professional software developers within Microsoft" (selection risk); "We analyze the variation in productivity perceptions based on an online survey with 413 professional software developers at Microsoft" (conflicting interests risk, due to the affiliation of an author);
(MoazeniLCB14) [41]	"The threat is mitigated for professional and student developers by the likelihood of distortions being common to all parts of the project" (measurement risk); "For a limited range of increments within a minor version of projects that have been going on for many years, the staff size and the applied effort of the staff members remained either constant or did not change significantly" (observation risk);
(Mockus09) [59]	"We investigate software development at Avaya with many past and present projects of various sizes and types involving more than 2000 developers" (conflicting interests risk, studied developers affiliated to the employer of the author);
(MosesFPS06) [51]	"It is necessary to assume that SLOC are counted in approximately the same way for the company" (measurement risk);
(OliveiraEA20) [46]	"We have contacted as many companies as possible to ask for authorization to analyze their projects" (inception risk);
(PalaciosCSGT14) [42]	"Participants were obtained from those who responded positively to a personal invitation sent by the authors to contacts working in Spanish and French IT companies" (selection-availability risk);
(PortM99) [69]	"The organization requesting the study hoped to compare the projects through the metric of productivity", "The customer of this study was particularly interested in this aspect" (conflicting interests risk, studied projects supported by the employer of an author);
(PotokV97) [68]	"The empirical data was collected at the IBM Software Solutions Laboratory in Research Triangle Park, North Carolina" (conflicting interests risk, studied projects supported by the employer of the authors);
(PotokVR99) [48]	"The empirical data discussed in this paper was collected at IBM Software Solutions", "The measurements collected are defined by a corporate metric council" (conflicting interests risk, studied projects supported by the employer of an author);
(PremrajSKF05) [50]	"The authors regret that presently the data set is not publicly available" (conflicting interests risk, studied database supported by the employer of an author);
(RamasubbuCBH11) [38]	"CodeMine provides a data collection framework for all major Microsoft development teams", "We conducted quantitative analysis on the version control system data and employee information stores in CodeMine" (conflicting interests risk, due to the affiliation of most authors);
(SiokT07) [72]	"The goal of this study was to provide answers to several questions regarding software development productivity and product quality within the avionics software engineering organization" (conflicting interests risk, studied projects supported by the employer of an author);
(StoreyEA21) [47]	"Our case company, Microsoft, is a large software company with tens of thousands of developers distributed in offices around the word" (conflicting interests risk, due to the affiliation of most authors);
(BissiNE16) [25]	No risk of bias assessment (performance risk);
(CardozoNBFS10) [26]	No risk of bias assessment (performance risk); Synthesis methods were not sufficiently detailed (selective non-reporting risk);
(HernandezLopezPG13) [8]	No risk of bias assessment (performance risk); Synthesis methods were not sufficiently detailed (selective non-reporting risk);
(MohagheghiC07) [18]	Paper screening, inclusion and exclusion criteria not sufficiently detailed (selection risk); No risk of bias assessment (performance risk);
(OliveiraVCC17) [27]	No risk of bias assessment (performance risk); Synthesis methods were not sufficiently detailed (selective non-reporting risk);
(OliveiraCCV18) [28]	No risk of bias assessment (performance risk); Synthesis methods were not sufficiently detailed (selective non-reporting risk);
(Peter11) [29]	No risk of bias assessment (performance risk);
(ShahPN15) [30]	No risk of bias assessment (performance risk); Synthesis methods were not sufficiently detailed (selective non-reporting risk);
(WagnerR08) [2]	"We inspected the first 100 results of each portal. We also collected papers manually in a number of important journals" (selection risk); No risk of bias assessment (performance risk); "The ProdFLOW method uses interview techniques for determining the most influential factors in productivity for a specific organization. ProdFLOW is a registered trademark of the Siemens AG" (conflicting interest risk, due to the affiliation of an author).

Appendix C. Evaluation of Certainty in the Body of Evidence

See in Table A3 the evaluation of certainty in the findings of each included study.

Table A3. Evaluation of Certainty in the Body of Evidence.

Key	Certainty Evaluation Criteria		
	C1	C2	C3
(AbdelHamid96) [79]	Low	Low	Low
(AdamsCB09) [36]	Low	Low	Low
(AsmildPK06) [70]	High	Low	High
(AzzehN17) [85]	High	Low	High
(AzzehN18) [86]	High	Low	High
(BankerDK91) [96]	High	Low	High
(BankerK91) [62]	Moderate	Unclear	Low
(BankerS94) [63]	Moderate	Low	Moderate
(BellerOBZ21) [74]	Moderate	Unclear	Low
(BeskerMB19) [87]	Moderate	Unclear	Low
(BezerraEA20) [88]	Moderate	Low	Moderate
(BibiAS16) [98]	Low	Low	Low
(BibiSA08) [91]	Moderate	Low	Moderate
(Boehm87) [21]	Moderate	Low	Moderate
(Boehm99a) [80]	Moderate	Low	Moderate
(CarvalhoRSCB11) [58]	Low	Unclear	Very low
(CataldoH13) [40]	Low	Low	Low
(ChapettaT20) [24]	Moderate	Low	Moderate
(Chatman95) [65]	Low	Unclear	Very low
(CheikhiARI12) [11]	Moderate	Low	Moderate
(DamianC06) [9]	Moderate	Low	Moderate
(DiesteEtAll17) [23]	High	Unclear	Moderate
(Duarte17a) [54]	Moderate	Low	Moderate
(Duncan88) [61]	Low	Unclear	Very low
(FatemaS17) [6]	Moderate	Low	Moderate
(FaulkLVSV09) [83]	Moderate	Unclear	Low
(FrakesS01) [81]	Moderate	Low	Moderate
(GeH11) [90]	Moderate	Low	Moderate
(GraziotinWA15) [4]	Moderate	Unclear	Low
(GreenHC05) [82]	Moderate	Low	Moderate
(HenshawJMB96) [66]	Low	Low	Low
(HernandezLopezCSC15) [7]	Moderate	Unclear	Low
(HernandezLopezPGC11) [32]	Moderate	Low	Moderate
(HuangW09) [99]	High	Low	High
(JaloteK21) [75]	Moderate	Unclear	Low
(JohnsonZB21) [76]	Moderate	Unclear	Low
(KautzJU14) [73]	Low	Low	Low
(KemayelMO91) [109]	Moderate	Low	Moderate
(KitchenhamM04) [22]	High	Low	High
(KreinMKDE10) [94]	High	Low	High
(KuutilaMCEA21) [102]	Moderate	Low	Moderate
(LagerstromWHL12) [57]	High	Low	High
(LavazzaMT18) [56]	Moderate	Low	Moderate
(LavazzaLM20) [93]	Moderate	Low	Moderate
(LiaoEA21) [95]	Moderate	Low	Moderate
(Lim94) [64]	Low	Unclear	Very low
(LowJ91) [77]	Moderate	Low	Moderate
(MacCormackKCC03) [35]	Moderate	Unclear	Low
(MantylaADGO16) [60]	Moderate	Low	Moderate
(Maxwe96) [108]	High	Low	High
(MaxwellF00) [49]	Low	Low	Low
(MeloCKC13) [12]	Low	Low	Low
(MeyerBMZF17) [5]	Moderate	Low	Moderate
(MeyerZF17) [10]	Moderate	Unclear	Low
(MinetakiM09) [104]	Moderate	Low	Moderate
(MoazeniLCB14) [41]	Low	Unclear	Very low
(Mockus09) [59]	High	Unclear	Moderate
(Mohapatra11) [37]	Moderate	Low	Moderate
(MosesFPS06) [51]	Low	Low	Low
(MurphyHillEA21) [89]	High	Low	High
(OliveiraEA20) [46]	Moderate	Low	Moderate
(PalaciosCSGT14) [42]	Moderate	Low	Moderate
(ParrishSHH04) [97]	Low	Low	Low
(PortM99) [69]	Low	Low	Low
(PotokV97) [68]	Low	Unclear	Very low
(PotokVR99) [48]	High	Low	High
(PremrajSKF05) [50]	High	Low	High
(RamasubbuCBH11) [38]	High	Low	High
(RastogiT0NC17) [45]	Moderate	Low	Moderate
(RodriguezSGH12) [39]	High	Low	High
(Rubin93a) [78]	Moderate	Low	Moderate
(Scacchi91) [103]	Moderate	Low	Moderate
(ScholtesMS16) [43]	High	Low	High

Table A3. *Cont.*

Key	Certainty Evaluation Criteria		
	C1	C2	C3
(SentasASB05) [92]	Moderate	Low	Moderate
(SiokT07) [72]	Moderate	Low	Moderate
(SovaS96) [67]	Low	Low	Low
(StaplesEA14) [101]	Moderate	Low	Moderate
(StoreyEA21) [47]	High	Low	High
(StylianouA16) [44]	Low	Low	Low
(Tan09) [84]	Low	Low	Low
(TanihanaN13) [100]	Low	Low	Low
(TomaszewskiL06) [71]	Low	Low	Low
(TrendM09) [105]	Moderate	Low	Moderate
(Tsuno09) [53]	Moderate	Low	Moderate
(TsunodaA17) [55]	Moderate	Low	Moderate
(Wang12) [106]	Moderate	Low	Moderate
(WangWZ08) [52]	Low	Low	Low
(YilmazOC16) [3]	Moderate	Low	Moderate
(ZhaoWW21) [107]	Moderate	Low	Moderate
(BissiNE16) [25]	High	Low	High
(CardozoNBFS10) [26]	High	Unclear	Moderate
(HernandezLopezPG13) [8]	High	Unclear	Moderate
(MohagheghiC07) [18]	High	Unclear	Moderate
(OliveiraVCC17) [27]	High	Unclear	Moderate
(OliveiraCCV18) [28]	High	Unclear	Moderate
(Peter11) [29]	High	Low	High
(RafiqueM13) [17]	High	Low	High
(ShahPN15) [30]	High	Unclear	Moderate
(WagnerR08) [2]	High	Unclear	Moderate

References

1. Boehm, B.W. *Software Engineering Economics*; Prentice-Hall: Hoboken, NJ, USA, 1981.
2. Wagner, S.; Ruhe, M. A Systematic Review of Productivity Factors in Software Development. In Proceedings of the 2nd International Workshop on Software Productivity Analysis and Cost Estimation (SPACE 2008), Beijing, China, 2 December 2008.
3. Yilmaz, M.; O'Connor, R.V.; Clarke, P. Effective Social Productivity Measurements during Software Development—An Empirical Study. *J. Softw. Eng. Knowl. Eng.* **2016**, *26*, 457–490. [CrossRef]
4. Graziotin, D.; Wang, X.; Abrahamsson, P. Do feelings matter? On the correlation of affects and the self-assessed productivity in software engineering. *J. Softw. Evol. Process* **2015**, *27*, 467–487. [CrossRef]
5. Meyer, A.; Barton, L.; Murphy, G.C.; Zimmermann, T.; Fritz, T. The Work-Life of Developers: Activities, Switches and Perceived Productivity. *IEEE Trans. Softw. Eng.* **2017**, *43*, 1178–1193. [CrossRef]
6. Fatema, I.; Sakib, K. Factors Influencing Productivity of Agile Software Development Teamwork: A Qualitative System Dynamics Approach. In Proceedings of the 24th Asia-Pacific Software Engineering Conference (APSEC 2017), Nanjing, China, 4–8 December 2017; Lv, J., Zhang, H.J., Hinchey, M., Liu, X., Eds.; IEEE: Piscataway, NJ, USA, 2017; pp. 737–742.
7. Hernández-López, A.; Palacios, R.C.; Soto-Acosta, P.; Casado-Lumbreras, C. Productivity Measurement in Software Engineering: A Study of the Inputs and the Outputs. *Int. J. Inf. Technol. Syst. Appl.* **2015**, *8*, 46–68. [CrossRef]
8. Hernández-López, A.; Palacios, R.C.; García-Crespo, Á. Software Engineering Job Productivity—A Systematic Review. *J. Softw. Eng. Knowl. Eng.* **2013**, *23*, 387–406. [CrossRef]
9. Damian, D.; Chisan, J. 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 Trans. Softw. Eng.* **2006**, *32*, 433–453. [CrossRef]
10. Meyer, A.; Zimmermann, T.; Fritz, T. Characterizing Software Developers by Perceptions of Productivity. In Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM 2017), Markham, ON, Canada, 9–10 November 2017; Bener, A., Turhan, B., Biffl, S., Eds.; IEEE: Piscataway, NJ, USA, 2017; pp. 105–110.
11. Cheikhi, L.; Al-Qutaish, R.E.; Idri, A. Software Productivity: Harmonization in ISO/IEEE Software Engineering Standards. *J. Softw.* **2012**, *7*, 462–470. [CrossRef]
12. de O. Melo, C.; Cruzes, D.S.; Kon, F.; Conradi, R. Interpretative case studies on agile team productivity and management. *Inf. Softw. Technol.* **2013**, *55*, 412–427. [CrossRef]
13. Duarte, C.H.C. The Quest for Productivity in Software Engineering: A Practitioners Systematic Literature Review. In Proceedings of the International Conference of Systems and Software Processes (ICSSP 2019), Montreal, QC, Canada, 25 May 2019; pp. 145–154.
14. Kitchenham, B.; Charters, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*; Technical Report EBSE 2007-001, Keele University and Durham University Joint Report. 2007. Available online: https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf (accessed on 31 January 2022).
15. Page, M.J.; McKenzie, J.E.; Bossuyt, P.M.; Boutron, I.; Hoffmann, T.C.; Mulrow, C.D.; Shamseer, L.; Tetzlaff, J.M.; Akl, E.A.; Brennan, S.E.; et al. The PRISMA 2020 Statement: An Updated Guideline for Reporting Systematic Reviews. *PLoS Med.* **2021**, *18*, e1003583. [CrossRef]

16. Schünemann, H.; Brożek, J.; Guyatt, G.; Oxman, A. Handbook for Grading the Quality of Evidence and the Strength of Recommendations Using the GRADE Approach. 2013. Available online: https://gradepro.org/handbook (accessed on 31 January 2022).
17. Rafique, Y.; Misic, V.B. The Effects of Test-Driven Development on External Quality and Productivity: A Meta-Analysis. *IEEE Trans. Softw. Eng.* **2013**, *39*, 835–856. [CrossRef]
18. Mohagheghi, P.; Conradi, R. Quality, productivity and economic benefits of software reuse: A review of industrial studies. *Empir. Softw. Eng.* **2007**, *12*, 471–516. [CrossRef]
19. Basili, V.; Caldiera, G.; Rombach, H.D. Goal Question Metric (GCM) Approach. In *Encyclopedia of Software Engineering*; Wiley: Hoboken, NJ, USA, 2002; pp. 528–532.
20. Ley, M. DBLP: Some Lessons Learned. *Proc. VLDB Endow.* **2009**, *2*, 1493–1500. [CrossRef]
21. Boehm, B.W. Improving Software Productivity. *IEEE Comput.* **1987**, *20*, 43–57. [CrossRef]
22. Kitchenham, B.; Mendes, E. Software productivity measurement using multiple size measures. *IEEE Trans. Softw. Eng.* **2004**, *30*, 1023–1035. [CrossRef]
23. Dieste, O.; Aranda, A.M.; Uyaguari, F.U.; Turhan, B.; Tosun, A.; Fucci, D.; Oivo, M.; Juristo, N. Empirical evaluation of the effects of experience on code quality and programmer productivity: An exploratory study. *Empir. Softw. Eng.* **2017**, *22*, 2457–2542. [CrossRef]
24. Chapetta, W.A.; Travassos, G.H. Towards an evidence-based theoretical framework on factors influencing the software development productivity. *Empir. Softw. Eng.* **2020**, *25*, 3501–3543. [CrossRef]
25. Bissi, W.; Neto, A.G.S.S.; Emer, M.C.F.P. The effects of test-driven development on internal quality, external quality and productivity: A systematic review. *Inf. Softw. Technol.* **2016**, *74*, 45–54. [CrossRef]
26. Cardozo, E.S.F.; Neto, J.B.F.A.; Barza, A.; França, A.C.C.; da Silva, F.Q.B. Scrum and Productivity in Software Projects: A Systematic Literature Review. In Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering (EASE 2010), Keele, UK, 12–13 April 2010.
27. de Oliveira, E.C.C.; Viana, D.; Cristo, M.; Conte, T. How have Software Engineering Researchers been Measuring Software Productivity? A Systematic Mapping Study. In Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017), Porto, Portugal, 26–29 April 2017; Hammoudi, S., Smialek, M., Camp, O., Filipe, J., Eds.; SciTePress: Setubal, Portugal, 2017; Volume 2, pp. 76–87.
28. de Oliveira, E.C.C.; Conte, T.; Cristo, M.; Valentim, N.M.C. Influence Factors in Software Productivity: Tertiary Literature Review. *J. Softw. Eng. Knowl. Eng.* **2018**, *28*, 1795–1810. [CrossRef]
29. Petersen, K. Measuring and Predicting Software Productivity. *Inf. Softw. Technol.* **2011**, *53*, 317–343. [CrossRef]
30. Shah, S.M.A.; Papatheocharous, E.; Nyfjord, J. Measuring productivity in agile software development process: A scoping study. In Proceedings of the International Conference on Software and System Process (ICSSP 2015), Tallinn, Estonia, 24–26 August 2015; pp. 102–106.
31. Jalali, S.; Wohlin, C. Systematic Literature Studies: Database Searches vs. Backward Snowballing. In Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM 2012), Lund, Sweden, 19–20 September 2012; pp. 29–38.
32. Hernández-López, A.; Palacios, R.C.; García-Crespo, Á.; Cabezas-Isla, F. Software Engineering Productivity: Concepts, Issues and Challenges. *Int. J. Inf. Technol. Syst. Approach* **2011**, *2*, 37–47. [CrossRef]
33. Bourque, P.; Fairley, R.E. (Eds.) *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, 3rd ed.; IEEE: Piscataway, NJ, USA, 2014.
34. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2012.
35. MacCormack, A.; Kemerer, C.F.; Cusumano, M.A.; Crandall, B. Trade-offs between Productivity and Quality in Selecting Software Development Practices. *IEEE Softw.* **2003**, *20*, 78–85. [CrossRef]
36. Adams, P.J.; Capiluppi, A.; Boldyreff, C. Coordination and productivity issues in free software: The role of Brooks' Law. In Proceedings of the 25th International Conference on Software Maintenance (ICSM 2009), Edmonton, AB, Canada, 20–26 September 2009; pp. 319–328.
37. Mohapatra, S. Maximising productivity by controlling influencing factors in commercial software development. *J. Inf. Commun. Technol.* **2011**, *3*, 160–179. [CrossRef]
38. Ramasubbu, N.; Cataldo, M.; Balan, R.K.; Herbsleb, J.D. Configuring global software teams: A multi-company analysis of project productivity, quality, and profits. In Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011), Honolulu, HI, USA, 21–28 May 2011; Taylor, R.N., Gall, H.C., Medvidovic, N., Eds.; ACM: New York, NY, USA, 2011; pp. 261–270.
39. Rodríguez-García, D.; Sicilia, M.; Barriocanal, E.G.; Harrison, R. Empirical findings on team size and productivity in software development. *J. Syst. Softw.* **2012**, *85*, 562–570. [CrossRef]
40. Cataldo, M.; Herbsleb, J.D. Coordination Breakdowns and Their Impact on Development Productivity and Software Failures. *IEEE Trans. Softw. Eng.* **2013**, *39*, 343–360. [CrossRef]
41. Moazeni, R.; Link, D.; Chen, C.; Boehm, B.W. Software domains in incremental development productivity decline. In Proceedings of the International Conference on Software and Systems Process (ICSSP 2014), Nanjing, China, 26–28 May 2014; Zhang, H., Huang, L., Richardson, I., Eds.; ACM: New York, NY, USA, 2014; pp. 75–83.

42. Palacios, R.C.; Casado-Lumbreras, C.; Soto-Acosta, P.; García-Peñalvo, F.J.; Tovar, E. Project managers in global software development teams: A study of the effects on productivity and performance. *Softw. Qual. J.* **2014**, *22*, 3–19. [CrossRef]
43. Scholtes, I.; Mavrodiev, P.; Schweitzer, F. From Aristotle to Ringelmann: A large-scale analysis of team productivity and coordination in Open-Source Software projects. *Empir. Softw. Eng.* **2016**, *21*, 642–683. [CrossRef]
44. Stylianou, C.; Andreou, A.S. Investigating the impact of developer productivity, task interdependence type and communication overhead in a multi-objective optimization approach for software project planning. *Adv. Eng. Softw.* **2016**, *98*, 79–96. [CrossRef]
45. Rastogi, A.; Thummalapenta, S.; Zimmermann, T.; Nagappan, N.; Czerwonka, J. Ramp-up Journey of New Hires: Do strategic practices of software companies influence productivity? In Proceedings of the 10th Innovations in Software Engineering Conference (ISEC 2017), Jaipur, India, 5–7 February 2017; pp. 107–111.
46. Oliveira, E.; Fernandes, E.; Steinmacher, I.; Cristo, M.; Conte, T.; Garcia, A. Code and commit metrics of developer productivity: A study on team leaders perceptions. *Empir. Softw. Eng.* **2020**, *25*, 2519–2549. [CrossRef]
47. Storey, M.A.D.; Zimmermann, T.; Bird, C.; Czerwonka, J.; Murphy, B.; Kalliamvakou, E. Towards a Theory of Software Developer Job Satisfaction and Perceived Productivity. *IEEE Trans. Softw. Eng.* **2021**, *47*, 2125–2142. [CrossRef]
48. Potok, T.E.; Vouk, M.A.; Rindos, A. Productivity Analysis of Object-Oriented Software Development in a Commercial Environment. *Softw. Pract. Exp.* **1999**, *29*, 833–847. [CrossRef]
49. Maxwell, K.D.; Forselius, P. Benchmarking Software-Development Productivity. *IEEE Softw.* **2000**, *17*, 80–88. [CrossRef]
50. Premraj, R.; Shepperd, M.J.; Kitchenham, B.A.; Forselius, P. An Empirical Analysis of Software Productivity over Time. In Proceedings of the 11th International Symposium on Software Metrics (METRICS 2005), Como, Italy, 19–22 September 2005; pp. 37–46.
51. Moses, J.; Farrow, M.; Parrington, N.; Smith, P. A productivity benchmarking case study using Bayesian credible intervals. *Softw. Qual. J.* **2006**, *14*, 37–52. [CrossRef]
52. Wang, H.; Wang, H.; Zhang, H. Software Productivity Analysis with CSBSG Data Set. In Proceedings of the International Conference on Computer Science and Software Engineering (CSSE 2008), Wuhan, China, 12–14 December 2008; Volume 2, pp. 587–593.
53. Tsunoda, M.; Monden, A.; Yadohisa, H.; Kikuchi, N.; Matsumoto, K. Software Development Productivity of Japanese Enterprise Applications. *Inf. Technol. Manag.* **2009**, *10*, 193–205. [CrossRef]
54. Duarte, C.H.C. Productivity Paradoxes Revisited: Assessing the Relationship Between Quality Maturity Levels and Labor Productivity in Brazilian Software Companies. *Empir. Softw. Eng.* **2017**, *22*, 818–847. [CrossRef]
55. Tsunoda, M.; Amasaki, S. On Software Productivity Analysis with Propensity Score Matching. In Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM 2017), Toronto, ON, Canada, 9–10 November 2017; Bener, A., Turhan, B., Biffl, S., Eds.; IEEE: Piscataway, NJ, USA, 2017; pp. 436–441.
56. Lavazza, L.; Morasca, S.; Tosi, D. An Empirical Study on the Factors Affecting Software Development Productivity. *e-Inform. Softw. Eng. J.* **2018**, *12*, 27–49.
57. Lagerström, R.; von Würtemberg, L.M.; Holm, H.; Luczak, O. Identifying factors affecting software development cost and productivity. *Softw. Qual. J.* **2012**, *20*, 395–417. [CrossRef]
58. de Souza Carvalho, W.C.; Rosa, P.F.; dos Santos Soares, M.; da Cunha, M.A.T., Jr.; Buiatte, L.C. A Comparative Analysis of the Agile and Traditional Software Development Processes Productivity. In Proceedings of the 30th International Conference of the Chilean Computer Science Society (SCCC 2011), Curico, Chile, 9–11 November 2011; pp. 74–82.
59. Mockus, A. Succession: Measuring transfer of code and developer productivity. In Proceedings of the 31st International Conference on Software Engineering (ICSE 2009), Vancouver, BC, Canada, 16–24 May 2009; pp. 67–77.
60. Mantyla, M.; Adams, B.; Destefanis, G.; Graziotin, D.; Ortu, M. Mining Valence, arousal, and Dominance—Possibilities for detecting burnout and productivity? In Proceedings of the 13th Conference on Mining Software Repositories (MSR 2016), Austin, TX, USA, 14–22 May 2016; Kim, M., Robbes, R., Bird, C., Eds.; ACM: New York, NY, USA, 2016; pp. 247–258.
61. Duncan, A.S. Software Development Productivity Tools and Metrics. In Proceedings of the 10th International Conference on Software Engineering (ICSE 1988), Singapore, 11–15 April 1988; Nam, T.C., Druffel, L.E., Meyer, B., Eds.; IEEE: Piscataway, NJ, USA, 1988; pp. 41–48.
62. Banker, R.D.; Kauffman, R.J. Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study. *MIS Q.* **1991**, *15*, 375–401. [CrossRef]
63. Banker, R.D.; Slaughter, S. Project Size and Software Maintenance Productivity: Empirical Evidence on Economies of Scale in Software Maintenance. In Proceedings of the 15th International Conference on Information Systems, Vancouver, BC, Canada, 14–17 December 1994; DeGross, J.I., Huff, S.L., Munro, M., Eds.; Association for Information Systems: Atlanta, GA, USA, 1994; pp. 279–289.
64. Lim, W.C. Effects of Reuse on Quality, Productivity, and Economics. *IEEE Softw.* **1994**, *11*, 23–30. [CrossRef]
65. Chatman, V.V., III. CHANGE-POINTs: A proposal for software productivity measurement. *J. Syst. Softw.* **1995**, *31*, 71–91. [CrossRef]
66. Bruckhaus, T.; Madhavji, N.H.; Henshaw, J.; Janssen, I. The Impact of Tools on Software Productivity. *IEEE Softw.* **1996**, *13*, 29–38. [CrossRef]
67. Sova, D.W.; Smidts, C.S. Increasing testing productivity and software quality: A comparison of software testing methodologies within NASA. *Empir. Softw. Eng.* **1996**, *1*, 165–188. [CrossRef]

68. Potok, T.E.; Vou, M.A. The Effects of the Business Model on Object-Oriented Software Development Productivity. *IBM Syst. J.* **1997**, *36*, 140–161. [CrossRef]
69. Port, D.; McArthur, M. A Study of Productivity and Efficiency for Object-Oriented Methods and Languages. In Proceedings of the 6th Asia-Pacific Software Engineering Conference (APSEC 1999), Takamatsu, Japan, 7–10 December 1999; pp. 128–135.
70. Asmild, M.; Paradi, J.C.; Kulkarni, A. Using Data Envelopment Analysis in software development productivity measurement. *Softw. Process. Improv. Pract.* **2006**, *11*, 561–572. [CrossRef]
71. Tomaszewski, P.; Lundberg, L. The increase of productivity over time—An industrial case study. *Inf. Softw. Technol.* **2006**, *48*, 915–927. [CrossRef]
72. Siok, M.F.; Tian, J. Empirical Study of Embedded Software Quality and Productivity. In Proceedings of the 10th International Symposium on High-Assurance Systems Engineering (HASE 2007), Dallas, TX, USA, 14–16 November 2007; pp. 313–320.
73. Kautz, K.; Johansen, T.H.; Uldahl, A. The Perceived Impact of the Agile Development and Project Management Method Scrum on Information Systems and Software Development Productivity. *Australas. J. Inf. Syst.* **2014**, *18*, 303–315. [CrossRef]
74. Beller, M.; Orgovan, V.R.; Buja, S.; Zimmermann, T. Mind the Gap: On the Relationship between Automatically Measured and Self-Reported Productivity. *IEEE Softw.* **2021**, *38*, 24–31. [CrossRef]
75. Jalote, P.; Kamma, D. Studying Task Processes for Improving Programmer Productivity. *Trans. Softw. Eng.* **2021**, *47*, 801–817. [CrossRef]
76. Johnson, B.; Zimmermann, T.; Bird, C. The Effect of Work Environments on Productivity and Satisfaction of Software Engineers. *IEEE Trans. Softw. Eng.* **2021**, *47*, 736–757. [CrossRef]
77. Low, G.; Jeffery, D. Software development productivity and back-end CASE tools. *Inf. Softw. Technol.* **1991**, *33*, 616–621. [CrossRef]
78. Rubin, H.A. Software process maturity: Measuring its impact on productivity and quality. In Proceedings of the 15th International Conference on Software Engineering (ICSE 1993), Baltimore, MA, USA, 17–21 May 1993; pp. 468–476.
79. Abdel-Hamid, T.K. The Slippery Path to Productivity Improvement. *IEEE Softw.* **1996**, *13*, 43–52. [CrossRef]
80. Boehm, B.W. Managing Software Productivity and Reuse. *IEEE Comput.* **1999**, *32*, 111–113. [CrossRef]
81. Frakes, W.B.; Succi, G. An industrial study of reuse, quality, and productivity. *J. Syst. Softw.* **2001**, *57*, 99–106. [CrossRef]
82. Green, G.C.; Hevner, A.R.; Collins, R.W. The impacts of quality and productivity perceptions on the use of software process improvement innovations. *Inf. Softw. Technol.* **2005**, *47*, 543–553. [CrossRef]
83. Faulk, S.R.; Loh, E.; de Vanter, M.L.V.; Squires, S.; Votta, L.G. Scientific Computing's Productivity Gridlock: How Software Engineering Can Help. *Comput. Sci. Eng.* **2009**, *11*, 30–39. [CrossRef]
84. Tan, T.; Li, Q.; Boehm, B.; Yang, Y.; Hei, M.; Moazeni, R. Productivity Trends in Incremental and Iterative Software Development. In Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), Lake Buena Vista, FL, USA, 15–16 October 2009; pp. 1–10.
85. Azzeh, M.; Nassif, A.B. Analyzing the relationship between project productivity and environment factors in the use case points method. *J. Softw. Evol. Process* **2017**, *29–53*, e1882. [CrossRef]
86. Azzeh, M.; Nassif, A.B. Project productivity evaluation in early software effort estimation. *J. Softw. Evol. Process* **2018**, *30*, e2110. [CrossRef]
87. Besker, T.; Martini, A.; Bosch, J. Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work. *J. Syst. Softw.* **2019**, *156*, 41–61. [CrossRef]
88. Bezerra, C.I.M.; de Souza Filho, J.C.; Coutinho, E.F.; Gama, A.; Ferreira, A.L.; ao de Andrade, G.L.; Feitosa, C.E. How Human and Organizational Factors Influence Software Teams Productivity in COVID-19 Pandemic: A Brazilian Survey. In Proceedings of the 34th Brazilian Symposium on Software Engineering (SBES 2020), Natal, Brazil, 21–23 October 2020; pp. 606–615.
89. Murphy-Hill, E.R.; Jaspan, C.; Sadowski, C.; Shepherd, D.C.; Phillips, M.; Winter, C.; Knight, A.; Smith, E.K.; Jorde, M. What Predicts Software Developers' Productivity? *IEEE Trans. Softw. Eng.* **2021**, *47*, 582–594. [CrossRef]
90. Ge, C.; Huang, K. Productivity Differences and Catch-Up Effects among Software as a Service Firms: A Stochastic Frontier Approach. In Proceedings of the International Conference on Information Systems (ICIS 2011), Shanghai, China, 4–7 December 2011; Galletta, D.F., Liang, T., Eds.; Association for Information Systems: Atlanta, GA, USA, 2011.
91. Bibi, S.; Stamelos, I.; Ampatzoglou, A. Combining probabilistic models for explanatory productivity estimation. *Inf. Softw. Technol.* **2008**, *50*, 656–669. [CrossRef]
92. Sentas, P.; Angelis, L.; Stamelos, I.; Bleris, G.L. Software productivity and effort prediction with ordinal regression. *Inf. Softw. Technol.* **2005**, *47*, 17–29. [CrossRef]
93. Lavazza, L.; Liu, G.; Meli, R. Productivity of software enhancement projects: An empirical study. In Proceedings of the Joint 30th International Workshop on Software Measurement and the 15th International Conference on Software Process and Product Measurement (IWSM-Mensura 2020), Mexico City, Mexico, 29–30 October 2020.
94. Krein, J.L.; MacLean, A.C.; Knutson, C.D.; Delorey, D.P.; Eggett, D. Impact of Programming Language Fragmentation on Developer Productivity: A Sourceforge Empirical Study. *Int. J. Open-Source Softw. Process.* **2010**, *2*, 41–61. [CrossRef]
95. Liao, Z.; Zhao, Y.; Liu, S.; Zhang, Y.; Liu, L.; Long, J. The Measurement of the Software Ecosystem's Productivity with GitHub. *Comput. Syst. Sci. Eng.* **2021**, *36*, 239–258. [CrossRef]
96. Banker, R.D.; Datar, S.M.; Kemerer, C.F. Model to evaluate variables impact in the productivity of software maintenance projects. *Manag. Sci.* **1991**, *37*, 1–18. [CrossRef]

97. Parrish, A.S.; Smith, R.K.; Hale, D.P.; Hale, J.E. A Field Study of Developer Pairs: Productivity Impacts and Implications. *IEEE Softw.* **2004**, *21*, 76–79. [CrossRef]
98. Bibi, S.; Ampatzoglou, A.; Stamelos, I. A Bayesian Belief Network for Modeling Open-Source Software Maintenance Productivity. In Proceedings of the International Conference 12th IFIP WG 2.13 Open-Source Systems: Integrating Communities (OSS 2016), Gothenburg, Sweden, 30 May–2 June 2016; IFIP Advances in Information and Communication Technology; Springer: Berlin/Heidelberg, Germany, 2016; Volume 472, pp. 32–44.
99. Huang, K.; Wang, M. Firm-Level Productivity Analysis for Software as a Service Companies. In Proceedings of the Information Conference on Information Systems (ICIS 2009), Phoenix, AZ, USA, 15–18 December 2009; pp. 1–17.
100. Tanihana, K.; Noda, T. Empirical Study of the Relation between Open-Source Software Use and Productivity of Japan's Information Service Industries. In Proceedings of the 9th IFIP WG 2.13 International Conference on Open-Source Software: Quality Verification (OSS 2013), Koper-Capodistria, Slovenia, 25–28 June 2013; Petrinja, E., Succi, G., Joni, N.E., Sillitti, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 18–29.
101. Staples, M.; Jeffery, R.; Andronick, J.; Murray, T.; Klein, G.; Kolanski, R. Productivity for proof engineering. In Proceedings of the 8th International Symposium on Empirical Software Engineering and Measurement (ESEM 2014), Torino, Italy, 18–19 September 2014; Morisio, M., Dyba, T., Torchiano, M., Eds.; ACM: New York, NY, USA, 2014; pp. 1–4.
102. Kuutila, M.; Mäntylä, M.; Claes, M.; Elovainio, M.; Adams, B. Individual differences limit predicting well-being and productivity using software repositories: A longitudinal industrial study. *Empir. Softw. Eng.* **2021**, *26*, 88. [CrossRef]
103. Scacchi, W. Understanding Software Productivity: Towards a Knowledge-Based Approach. *J. Softw. Eng. Knowl. Eng.* **1991**, *1*, 293–321. [CrossRef]
104. Minetaki, K.; Motohashi, K. Subcontracting Structure and Productivity in the Japanese Software Industry. *Rev. Socionetw. Strateg.* **2009**, *3*, 51–65. [CrossRef]
105. Trendowicz, A.; Münch, J. Factors Influencing Software Development Productivity: State-of-the-Art and Industrial Experiences. *Adv. Comput.* **2009**, *77*, 185–241.
106. Wang, Y.; Zhang, C.; Chen, G.; Shi, Y. Empirical research on the total factor productivity of Chinese software companies. In Proceedings of the International Joint Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT 2012), Macau, China, 4–7 December 2012; Volume 3, pp. 25–29.
107. Zhao, L.; Wang, X.; Wu, S. The Total Factor Productivity of China's Software Industry and its Promotion Path. *IEEE Access* **2021**, *9*, 96039–96055. [CrossRef]
108. Maxwell, K.; Wassenhove, L.V.; Dutta, S. Software development productivity of European space, military and industrial applications. *IEEE Trans. Softw. Eng.* **1996**, *22*, 706–718. [CrossRef]
109. Kemayel, L.; Mili, A.; Ouederni, I. Controllable factors for programmer productivity: A statistical study. *J. Syst. Softw.* **1991**, *16*, 151–163. [CrossRef]
110. Budgen, D.; Brereton, P.; Drummond, S.; Williams, N. Reporting systematic reviews: Some lessons from a tertiary study. *Inf. Softw. Technol.* **2018**, *95*, 62–74. [CrossRef]
111. Higgins, J.P.T.; Thomas, J.; Chandler, J.; Cumpston, M.; Li, T.; Page, M.J.; Welch, V.A. (Eds.) *Cochrane Handbook of Systematic Reviews of Interventions*, 6.2 version; Wiley-Blackwell: Hoboken, NJ, USA, 2021.
112. McGuinness, L.A.; Higgins, J.P.T. Risk-of-bias Visualization (Robvis): An R package and Shiny web app for visualizing risk-of-bias assessments. *Res. Synth. Methods* **2021**, *12*, 55–61. [CrossRef]
113. Guyatt, G.; Oxman, A.D.; Akl, E.A.; Kunz, R.; Vist, G.; Brozek, J.; Norris, S.; Falck-Ytter, Y.; Glasziou, P.; DeBeer, H.; et al. GRADE Guidelines: 1. Introduction—GRADE Evidence Profiles and Summary of Findings Tables. *J. Clin. Epidemiol.* **2011**, *64*, 383–394. [CrossRef]
114. Brereton, P.; Kitchenham, B.A.; Budgen, D.; Turner, M.; Khalil, M. Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. *J. Syst. Softw.* **2007**, *80*, 571–583. [CrossRef]
115. Whiting, P.; Savović, J.; Higgins, J.P.; Caldwell, D.M.; Reeves, B.C.; Shea, B.; Davies, P.; Kleijnen, J.; Churchill, R. ROBIS: A new tool to assess risk of bias in systematic reviews was developed. *J. Clin. Epidemiol.* **2016**, *69*, 225–234. [CrossRef]