Journal of Software Engineering Research and Development, 2020, 8:8, DOI: 10.5753/jserd.2020.544

How is a Developer's Work Measured? An Industrial and Academic Exploratory View

Matheus Silva Ferreira [Federal University of Lavras | matheus.ferreira5@estudante.ufla.br]
Luana Almeida Martins [Federal University of Lavras | luana.martins1@estudante.ufla.br]
Paulo Afonso Parreira Júnior [Federal University of Lavras | pauloa.junior@ufla.br]
Heitor Costa [Federal University of Lavras | heitor@ufla.br]

Abstract

Software Project Management is an essential practice to successfully achieve goals in software development projects and a challenging task for Project Managers (PMs). Therefore, information about the developers' work can be valuable in supporting the PMs' activities. Several studies address this topic and suggest different strategies for obtaining such information. Given the variety of existing strategies, we need to know the state-of-the-art on the theme. This article presents the information used for supporting PMs in the application of project management practices, especially with regard to risk management and people management. Thus, we carried out an exploratory study using a Systematic Mapping Study (SMS). Contributions include the identification of 64 metrics, four information sources, and seven PM activities supported by the measurement of the developers' work. Additionally, we interviewed four PMs to collect their personal opinion of how the metrics and activities reported by our SMS could help the project management in practice. Each PM considered a different set of metrics to support their activities, but none of them suggested new metrics (besides the 64 metrics identified in the SMS). Also, we presented aspects to explore the subject, indicating themes for possible new studies in the Software Engineering area.

Keywords: *Project Management, Knowledge of the Developers' Work, Project Manager's Activities, Metric*

1 Introduction

In software projects, the Project Manager (PM) is the professional who ensures proper project management. The PM's function includes selecting members for the project team and assigning roles and responsibilities as needed (de Souza *et al.* 2015). PMs must know how to assess the skills, strengths, and weaknesses of developers so that they can do their work efficiently (Zuser and Grechenig 2003). Besides, bad people management can bring about project risks (Ferreira *et al.* 2017). For example, team member turnover can be a high risk for a project because some developers can centralize software source code knowledge (Boehm 1991). These issues correspond to the PMs' two activities: risk management and people management (Sommerville 2019).

Performing people management and risk management is a non-trivial task. In addition, the PM's effort is impacted by the size of the project and the size of the team (Ahonen *et al.* 2015). In this context, the team members' evaluation by PMs motivates studies in the literature (de Bassi *et al.* 2018; Feiner and Andrews 2018; Ferreira *et al.* 2017; Zuser and Grechenig 2003) that suggest strategies for measuring the developers' work. Given the wide variety of suggested strategies, we need to know the state-of-the-art approaches in this field.

This article presents an investigation on how the developers' work can be measured, and how information on the developers' work can support the project management (especially, risk management and people management). Thus, we performed an exploratory study using a Systematic Mapping Study (SMS). SMS allows to identify, interpret and evaluate available evidence from studies on a topic, phenomenon, or set of research questions of interest (Kitchenham 2004). A SMS has three phases (Kitchenham and Charters 2007): i) Planning (we define the motivation, goals, and research protocol); ii) Execution (we apply the strategy outlined in the research protocol to identify and select studies); e iii) Results (we show the analysis of information obtained from selected studies). Additionally, we interviewed four PMs to collect their opinion about results obtained in SMS that can help them in practice. Thus, we have an initial understanding of how the industry measures the developer's work.

The remainder of this article is organized as follows: Section 2 describes a theoretical framework. Section 3 presents the SMS Planning phase. Section 4 describes the SMS Execution phase. Section 5 presents the SMS Results phase. Section 6 discusses the results along with the PMs' opinion. Section 7 describes the threats to validity. Section 8 draws concluding remarks.

2 Background

This section discusses risk management and people management.

2.1 Risk Management

In PMBoK (Project Management Body of Knowledge), Risk Management is an area of knowledge that aims to identify, evaluate, and monitor the positive (opportunities) and negative (threats) risks that may affect the project (PMI 2017). The most critical activities for the PMs when a problem emerges are to evaluate and monitor risks (Sommerville 2019). This area comprises five processes

(Plan Risk Management, Identify Risks, Perform Qualitative Risk Analysis, Perform Quantitative Risk Analysis, and Plan Risk Responses). There are several techniques for analyzing threats. When choosing a threat analysis technique, the PM needs to pay attention to the characteristics of the project to avoid impacts on the quality of the analysis results (Tuma, Gül, and Riccardo 2018). Effective performance of risk management can directly impact project success or failure (Menezes *et al.* 2019).

The risks can affect the project schedule or resources (**project risks**), software quality or performance (**product risks**), or organization that produces or acquires software (**business risks**) (Sommerville 2019). For example, the departure of an experienced developer can represent:

- **Project risk**, because this departure affects the schedule due to the loss of human resources;
- **Product risk**, because the developer who substitutes an experienced developer can have different skills and project knowledge; and
- **Business risk**, because the developer's experience impacts the signing of contracts.

Many existent risks in software projects relate to the development team. Specifically, one risk addressed in the literature is the lack of technical skills of some team members (Menezes et al. 2019), leading to investments in training or hiring people. Another risk mentioned as a constant concern for PMs is developer turnover associated to concentration of knowledge regarding source codes. In this situation, the project can fail if these developers leave the project/organization earlier than expected (Ferreira et al. 2017). An alternative in order to mitigate this is to identify the people who concentrate the knowledge on source code and distribute it among all team members (developers).

2.2 People Management

In PMBoK, Resource Management is a field of knowledge that aims to identify, acquire, and manage the resources needed for successful project completion (PMI 2017). This area comprises six processes (Planning of Resource Management, Estimate Activity Resources, Acquire Resources, Develop Team, Manage Team, and Control Resources).

In software projects, the team members play different roles. Thus, PMs need to consider the members' technical skills and personality to assemble the teams. In order to correctly manage people, PMs should (Sommerville 2019):

- Have an honest and respectful relationship with those involved in the project;
- Have people who are motivated to perform their functions;
- Support teamwork and maintain relationship of trust among everyone, enabling the team to self-manage;
- Select team members to optimize performance and meet the projects' technical and human requirements;
- Organize the working method and team members' roles; and
- Ensure effective communication between the people involved.

Despite the existing recommendations, there are studies that show that PMs can hardly organize performance teams in a systematic and repeatable way (Latorre and Javier 2017). Understanding people's characteristics, assigning tasks, and recognizing the work done are complex and relevant tasks for PMs (Zuser and Grechenig 2003).

3 SMS Planning Phase

This section describes the SMS Planning phase and presents the research protocol and its validation and the data extraction procedure.

3.1 Research Protocol

The research protocol includes the strategies used for retrieving and selecting studies that are relevant to the topic of interest in the research (Kitchenham 2004). In this protocol, we defined the research questions, the procedure used to conduct SMS, the inclusion and exclusion criteria for selecting the studies, and how to obtain and classify information. In Table 1, we showed the research questions and the goals for answering them. The primary research questions help to understand the measurement of the developers' work to support PMs and consist of the main results of this study. The secondary research questions provide insight into the characteristics of the scientific studies found in the SMS.

We selected the ACM, IEEE, and Springer repositories of scientific papers. In addition, we chose Ei Compendex and Scopus because they index other repositories. They publish papers from the most important conferences and journals in Software Engineering (Ambreen *et al.*, 2018; Bouchkira 2020). Additionally, we elaborated on a search string (Table 2) that contains five parts of key terms aligned with the research questions to retrieve studies in these repositories. Each set is composed of a key expression and its synonyms, as follows:

- **Part 1** refers to the action required to obtain the metrics on the developers' work. We defined it by *measure OR measurement OR mensuration OR dimension OR evaluation OR analyze OR analysis OR view OR visualization OR knowledge*;
- **Part 2** refers to the object to be measured (the developer's work). We defined it by *contribution OR participation OR productivity OR skills OR collaboration OR effort OR knowledge*;
- **Part 3** refers to who is evaluated by the measurement. We defined it by *developers OR "software development team" OR "team members"*;
- **Part 4** refers to whoever is interested in the metrics obtained from the measurement. We defined it by *"software project manager" OR "project manager" OR "project managers" OR "software project" OR "project management"*; and

- **Part 5** refers to the method or tool used to measure. We defined it by *tool OR framework OR plugin OR method OR metric OR factors*.

Only Part 1 was searched in the title of the studies because it is related to "measurement" and is relevant to the retrieved studies regarding the SMS goal (we used search engine tokens). When we enlarged the search for other items (besides the title), the number of returned studies increased significantly; these studies treat issues that are distant from the objective of this study. We searched the other parts (Parts 2 - 5) in titles, abstracts, and keywords.

Table 1. Research Questions

Primary Research Questions	
Research Question	**Goal**
Q-P.1 - What metrics are used by PMs to measure the developers' work?	To identify metrics about the developers' work.
Q-P.2 - How are metrics applied by PMs to monitor the developers' work?	To identify how PMs extract and analyze the metrics to measure the developers' work.
Q-P.3 - How do metrics concerning the developers' work support project management, especially with regard to risk management and people management?	To identify how metrics regarding the developers' work support project management (risk management and people management).
Secondary Research Questions	
Q-S.1 - What type of solution is often proposed for studies in this area?	To identify the type of solution proposed in studies to measure the developers' work and support the PMs' decision.
Q-S.2 - What type of research methodology is often used for studies in this area?	To identify the research methodology used in studies to verify their maturity.
Q-S.3 - How is the proposed solution related to the research methodology in the included studies?	To identify the maturity of the solutions presented in studies. For example, how the researchers evaluated the metrics used for measuring the developers' work.

Table 2. Search String

(measure OR measurement OR mensuration OR dimension OR evaluation OR analyze OR analysis OR view OR visualization)
AND *(contribution OR participation OR productivity OR skills OR collaboration OR effort OR knowledge)*
AND (*developers OR "software development team" OR "team members"*)
AND (*"software project manager" OR "project manager" OR "project managers" OR "software project" OR "project management"*)
AND (*tool OR framework OR plugin OR method OR metric OR factors*)

Next, we established the selection process, which defines the inclusion and exclusion criteria. For inclusion criteria, the study should be a primary study addressing the mensuration of developers' work to support the PMs' activities. For exclusion criteria, we removed studies that (i) do not have complete scientific contributions (*e.g.*, abstracts), (ii) are not scientific studies (*e.g.*, standards and tables of contents), (iii) do not have complete texts, or (iv) have restricted access.

Subsequently, a procedure that involved the efforts of four researchers was defined to select the studies. Researchers A and B performed the activities planned for SMS. Researcher C (experienced) helped Researchers A and B. Researcher D (the most experienced) supervised the work. The procedure consisted of the following stages:

- **To apply the search string.** Researcher A applied the search string in the digital repositories and stored the retrieved studies in the Mendeley reference management system (https://mendeley.com);
- **To remove duplicates**. Researchers A and B analyzed the information from the studies retrieved in the previous stage in order to identify and remove duplicate ones. They used a Mendeley feature to identify duplicate studies. They then removed indexed studies with fewer keywords because those with more keywords in the digital database can be considered as better characterized;
- **To apply exclusion criteria.** Researchers A and B applied the exclusion criteria. Researcher C monitored the exclusion;
- **To select potential studies**. Researchers A and B independently read the resulting studies' title, abstract, and keyword from the previous stage to identify those with the potential to meet the inclusion criteria. They classified them as "with potential", "without potential", or "doubtful" (unsure about the potential). Both researchers admitted the maximum of potential studies. Next, they merged their classification following two

decision criteria (**Accepted** or **Reject**) (Bin Ali and Petersen 2014). The researchers accepted a study if both classified it as "with potential" or if at least one classified it as "with potential". The researchers rejected the study if both classified it as "without potential" or if one researcher classified it as "without potential" and the other classified it as "doubtful". Researcher C monitored the classification process; and

- **To apply inclusion criteria**. Researchers A and B read the full text of the resulting studies from the previous stage and defined five quality questions for scoring the studies (Table 3) to apply the inclusion criteria. Each question could receive the value 1 (Yes), 0 (No), or 0.5 (Partly). Thus, the minimum score is 0 and the maximum score is 5. After assigning scores individually, Researchers A and B calculated the arithmetic mean for each study ((Researcher A score + Researcher B score)/2). This calculation represents the study's final score for the inclusion criteria. Finally, they accepted studies with a final score equal to or greater than 2.5 (50%) (Bin Ali and Petersen 2014). Researcher D analyzed the accepted studies to assess their relevance for SMS.

Table 3. Quality Questions for Inclusion Criteria

ID	Quality Questions
Q1	Are the aims of the research explicitly defined?
Q2	Are the metrics explicitly reported?
Q3	Are the metrics related to a software activity?
Q4	Are the metrics clearly described or defined?
Q5	Are the findings related to a project management activity?

3.2 Evaluation of the Research Protocol

We evaluated the research protocol before starting the Execution phase (Kitchenham and Charters 2007) to assess the feasibility of performing SMS and identifying the changes necessary in order to improve the quality of the retrieved studies. This evaluation occurred with one test that defined a group of primary studies (control group) to be retrieved by the research protocol. We set the group of control with 7 studies through an ad-hoc literature review, using Google Scholar to search for studies related to the SMS goal.

We refined the search string and applied it to the search engines until they returned all the studies from the control group. Keywords in Part 1 (Table 2) are general; the measurement process is commonly applied in the studies to validate their results. Therefore, we restricted Part 1 to be searched only in the title of the studies. Consequently, we found six studies from the control group; we added the other study manually. Table A1 - Appendix A lists the studies from the control group (P20 - protocol did not retrieve, and P21, P23, P24, P30, P33, and P39).

3.3 Data Classification and Extraction Procedure

We established a procedure to classify and extract data from the selected studies in nine categories:

- **Information on the publication.** To collect data related to the study's publication, such as title, authors, year of publication, and publication media (journal/event), and use them to track general information on the studies;
- **Proposal of Solution**. To collect data on the solution and classify it as "Overview", "Method", "Model", "Metric", and "Tool" (Petersen *et al.* 2008). This category helps to Q-S.1 and Q-S.3;
- **Research Methodology**. To collect data on the research methodology and classify it as "Evaluation Research", "Proposal of Solution", "Validation Research", "Opinion Studies", "Experience Studies", and "Philosophical Studies" (Wieringa *et al.* 2006). This category helps to answer Q-S.2 and Q-S.3;
- **Metrics to measure the developers' work.** To collect metrics used by PMs to measure the developers' work. This category helps to answer Q-P.1;
- **Data source on the developers' work.** To identify data sources used as input to apply metrics. This category helps to answer Q-P.2;
- **Method to extract data and apply metrics.** To identify the method used to obtain the metrics from the data sources and apply them (*e.g.*, mining of source code repositories and collecting feedback from team members). This category helps to answer Q-P.2;
- **Context of metrics extraction and application**. To collect metrics in a context: i) type of software (proprietary or open-source) and ii) environment to collect team data (academic or industry). This category helps to answer Q-P.2;
- **Method for presenting results to PMs.** To identify information about how to present metrics to PMs. This category helps to answer Q-P.2; and
- **Support for project management.** To identify how the metrics of the developers' work support PMs in project management, especially in risk management and people management. This category helps to answer Q-P.3.

4 SMS Execution Phase

We performed the SMS Execution Phase between March and November 2019. First, we customized and applied the search string in the selected search engines, considering their specificities. We used the search string in the:

- Single search field without filters (ACM Digital Library);
- Advanced Search field without filters (Scopus);
- Single search field with the filter "Content-Type: Conference Publications, Journals & Magazines" (IEEE Xplore);
- Single search field with the filter "Controlled Vocabulary: Software Engineering" (Ei Compendex);

- Single search field with the filters "Discipline: Computer Science; Subdiscipline: Software Engineering; Content-Type: Article" (Springer).

Next, we selected the studies, and the results were as follows (Table 4):

- **Application of the search string.** We retrieved 1,381 documents (Filter 1). Of these, 240 documents were from ACM (17.4%), 433 documents were from IEEE (31.4%), 169 documents were from Scopus (12.2%), 141 documents were from Ei Compendex (10.2%), and 398 documents were from Springer (28.8%);
- **Removal of duplicates.** After removing duplicate documents, 1,305 documents remained (Filter 2). Of these, 239 documents were from ACM (18.3%), 421 documents were from IEEE (32.3%), 115 documents were from Scopus (8.8%), 134 documents were from Ei Compendex (10.3%), and 396 documents were from Springer (30.3%);
- **Application of exclusion criteria.** After applying the exclusion criteria (after which only studies remained), 1,205 studies remained (Filter 3). Of these, 212 studies were from ACM (17.6%), 415 studies were from IEEE (34.4%), 77 studies were from Scopus (6.4%), 115 studies were from Ei Compendex (9.6%), and 386 studies were from Springer (32.0%). At this stage, the results returned were only studies;
- **Selection of potential studies.** After reading the titles of the studies, abstract, and keywords, 61 studies were retrieved (Filter 4). Of these, 7 studies were from ACM (11.5%), 32 studies were from IEEE (52.5%), 8 studies were from Scopus (13.1%), 8 studies were from Ei Compendex (13.1%), and 6 studies were from Springer (9.8%); and
- **Application of inclusion criteria**. After applying the inclusion criteria, we retrieved and read 40 studies. Additionally, we added the (only) non-recovered study (from the control group) by the protocol, totaling 41 accepted studies (Filter 5). Of these, 4 studies were from ACM (9.8%), 27 studies were from IEEE (65.9%), 6 studies were from Scopus (14.6%), 3 studies were from Ei Compendex (7.3%), and 1 study was from Springer (2.4%).

Table 4. Summary of Selection Stages

Repositories	Filter 1	Filter 2		Filter 3		Filter 4		Filter 5	
	A	A	R	A	R	A	R	A	R
ACM	240	239	1	212	27	7	205	4	3
IEEE	433	421	12	415	6	32	383	27	5
Scopus	169	115	54	77	38	8	69	6	2
Ei Compendex	141	134	7	115	19	8	107	3	5
Springer	398	396	2	386	10	6	380	1	5
Total	**1,381**	**1,305**	**76**	**1,205**	**100**	**61**	**1,144**	**41**[*]	**20**

* Included Studies from the Control Group

5 SMS Results Phase

The studies resulting from the SMS are presented in Table A1 (Appendix A), containing identifiers, authors, year of publication, and repositories. When extracting the data from the studies, we could observe that the date of the resulting studies began in 2003 and had a publication frequency (average) of 2 studies. Over the years, two periods had more publications on the subject (2008 and from 2014 to 2016), with an average of 5.5 studies published (Figure 1).

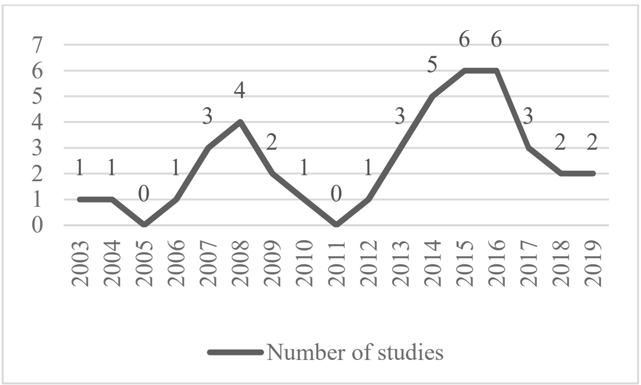

Figure 1. Annual Distribution of Selected Studies

Additionally, when analyzing the publication media, conferences and symposiums published 31 studies (75.6%), journals published 5 studies (12.2%), and workshops published 5 studies (12.2%). The International Conference on Software Maintenance and Evolution (ICSME) published more studies (5 studies).

To answer the secondary research question

Q-S.1 - What type of solution is often proposed for studies in this area?

we mapped the studies regarding the proposal of solutions to investigate the study theme, which could be (Figure 2):

- **Method** defines workflows, rules, or procedures on how to perform an activity. It was present in 9 studies (22.0%);
- **Model** describes conceptual representation with a formal abstraction of details and notations. It was present in 9 studies (22.0%);
- **Metric** describes new metrics or one measurement plan. It was present in 9 studies (22.0%);
- **Overview** describes and compares information to provide an overview of the subject. It was present in 4 studies (9.7%); and

- **Tool** describes and provides one computational tool. It was present in 10 studies (24.3%).

These solutions result from keywords often found in Software Engineering studies (Petersen *et al*. 2008). This mapping will allow us to assess where the community was more focused, i.e., on proposing new metrics or developing new tools to collect the existing metrics.

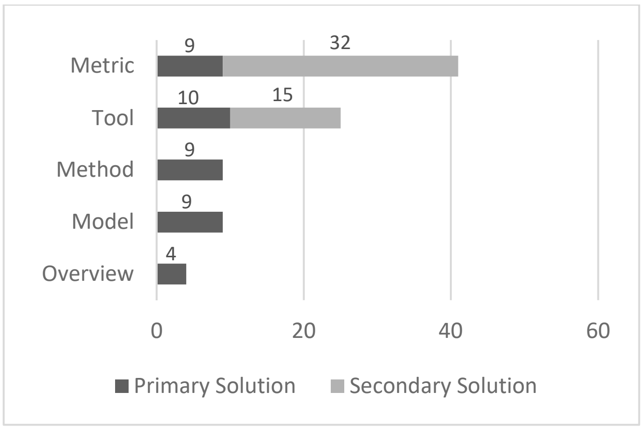

Figure 2. Proposal of Solutions

The identification of the solutions considers the first solution proposed in the study; the "Metric" and "Tool" solutions were also found as secondary solutions in other studies. All studies used the "Metric" solution. The "Tool" solution was used in 4 studies to support the "Metric" solution (26.7%), in 4 studies to support the "Method" solution (26.7%), 5 studies to support the "Model" solution (33.3%), and 2 studies to support the "Overview" solution (13.3%). In the following items, we discussed proposals for primary solutions:

- **Method** was used to mine information and knowledge to support collaborative programming and resource allocation. Workflow analysis and interaction among developers facilitate project understanding and development (P10, P16, P32, P37), considering collaborative programming. Historical details of resource allocation and individual skills were analyzed to distribute tasks to team members recognizing each developer's competencies (P1, P6, P11, P36), considering resource allocation;
- **Model** was used to analyze individual participation, which investigated the collaborative software engineering context in order to share information and organize tasks and resources. Hence, the identified models evaluated the developers' performance, considering the development environment and team feedback (P2, P7, P20), the building of the development team based on the activity, profile and experience of the developers (P4, P18, P34, P41), and the assessment of how the developers' roles evolved, based on their contributions (P19, P24);
- **Metric** was collected in source code repositories, bug tracking systems, or version control systems. From these metrics, indicators were taken from the developers' work, *e.g.*, productivity (P8, P9, P17, P21, P22, P23, P25), collaboration (P8, P9, P21, P22, P23), experience (P8, P9, P21, P22, P26), interaction (P8, P22), and task accomplishment indicators (P8, P9, P23, P25, P38);
- **Overview** consists in the investigation of how PMs understand the developers' work compared different factors regarding developers' personality and activity. PMs interviewed developers to identify the most appropriate profile for a task (P15, P33) and compared methods used to estimate the developers' work (P29, P35); and
- **Tool** was used to support PM activities. Using automated resource allocation in software projects can help PMs in analyzing the variables needed for resource allocation. PMs can examine the software development process through information extracted from software repositories (P3, P13, P14, P30, P39) and developers' evolution (P5, P12, P28, P30, P31, P40).

To answer the secondary research question

Q-S.2 - What type of research methodology is often used for studies in this area?

we mapped studies according to the research methodology used (Figure 3):

- **Proposal of Solution** proposes a solution technique and defends its relevance with one small example, or one good argumentation - 13 studies (31.7%) classified;
- **Validation Research** investigates a solution technique within a specific context through experiments, surveys, or interviews to answer a particular research question - 16 studies (39.0%) classified. This methodology does not require more formal experimental methods (*e.g.*, hypothesis testing, control experiment);
- **Evaluation Research** investigates the relationship among phenomena through formal experimental methods where casual properties are studied empirically, such as case studies, field studies, and field experiments - 12 studies (29.3%) classified;
- **Experience Studies** explains how something has been done in practice based on the author's experience - no study classified;
- **Opinion Studies** reports the author's opinion on how things should be - no study classified; and
- **Philosophical Studies** structures the information regarding a specific field like one specific taxonomy or conceptual framework, resulting in a new way of looking at existing things - no study classified.

We used three levels of research maturity (**high rigor**, **medium rigor**, and **low rigor**) related to the study subject (Garousi *et al.* 2015). The "Proposal of Solution" methodology has low rigor because it provides simple examples to verify its applicability. The "Validation Research" methodology has medium rigor because it does not include hypothesis testing nor discussions on threats to validity. The "Evaluation Research" methodology has high rigor because it includes hypothesis testing and discussions on threats to validity. Given the distribution of studies in these methodologies (Figure 3), there are more empirical studies than proposal of solution, indicating high research rigor in this area.

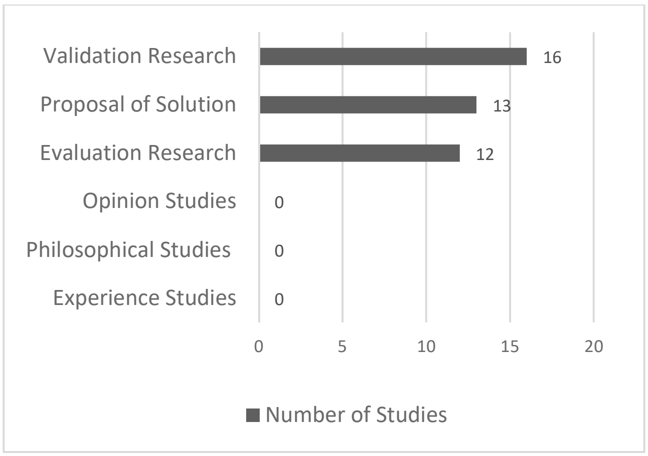

Figure 3. Research Methodology

To answer the secondary research question

Q-S.3 - How is the proposed solution related to the research methodology in the included studies?

we performed an integrated analysis of the results obtained in Q-S.2 and Q-S.3. The information on the number of studies by research methodology and type of primary solution were listed and presented in Figure 4. We can observe that many studies showed the "Metric", "Method", and "Model" methodologies as a solution in the "Proposal of Solution", "Validation Research", and "Evaluation Research" methodologies, i.e., they were evaluated in the three maturity levels (high, medium, and low rigor). However, many intersections have zero values. For example, in studies that presented the "Tool" solution, only evaluations with low and medium rigor were conducted using the "Proposal of Solution" and "Validation Survey" methodologies. Therefore, this solution needs robust empirical studies. Besides, there are no solutions related to the "Experience Studies", "Opinion Studies", and "Philosophical Studies" methodologies, which highlights the need for studies in this field.

To answer the primary research question

Q-P.1 - What metrics are used by PMs to measure the developers' work?

we collected 64 metrics used to measure the developers' work (Table 5) and categorized them into 6 groups (Figure 5). In this categorization, we considered the similarity between the meanings and purpose of the metrics. Thus, we merged two metrics with different names when their aims and values were the same. Besides, there were cases in which the metrics with the same name measured different information. In those cases, we separated them into two or more metrics and changed their names. One example was the Collaboration (files) and Collaboration (interaction) metrics. The first one relates to the joint work on code files, and the second one relates to the exchange of information and help by teammates. The categories are:

- **Quality (Qua).** This group refers to the quality of work delivered (task or source code). It covers the Martin (Martin 1994), CK (Chidamber and Kemerer 1994), size, and complexity metrics (P8, P9, P11, P14, P23, P25, P33, P39, P38). It comprises 11 metrics in 9 studies (22%);

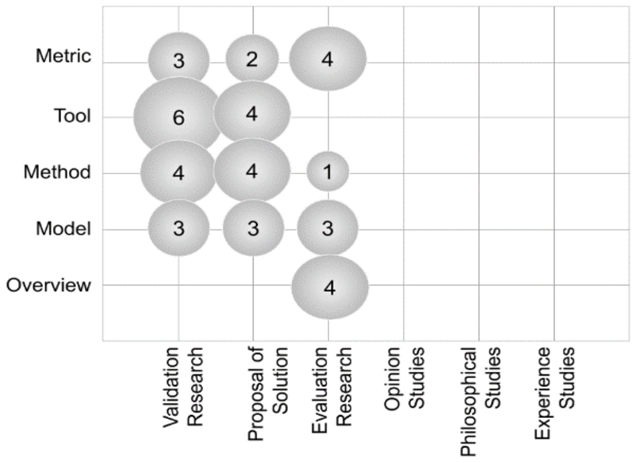

Figure 4. Relationship between Methodologies and Proposal of Solution

- **Contribution (Con).** This group refers to the amount of work performed by the developers on the software artifacts (P2, P3, P4, P5, P8, P9, P10, P11, P12, P13, P14, P16, P17, P18, P19, P20, P21, P22, P23, P24, P25, P26, P27, P28, P29, P30, P32, P33, P34, P35, P37, P39, P40). It covers metrics related to the number of commits and the number of modified/added/removed lines of code (Code Churn). It comprises 39 metrics in 33 studies (80.5%);
- **Collaborative Work (CoW).** This group refers to information sharing and teamwork to develop the same software artifact (P3, P7, P8, P9, P10, P12, P14, P18, P21, P22, P23, P30, P39, P40). It covers metrics related to the messages exchanged regarding bugs and the commits performing or lines of code in the same artifact. It comprises 4 metrics in 14 studies (34.1%);
- **Degree of Importance (ImP).** This group refers to technology domains, developer reputation, participation time in projects, and type of work (*e.g.*, bug fix) (P4, P5, P7, P8, P9, P13, P14, P15, P17, P18, P19, P20, P21, P22, P26, P27, P29, P31, P32, P33, P34, P35, P37). It comprises 20 metrics in 23 studies (56.1%).
- **Productivity (Pro).** This group refers to the number of tasks performed within a given time (P2, P8, P17, P20, P24, P25, P33, P34, P36). It relates to the number of modified/added/removed lines of code, the number of commits, or the report on tasks completed in a period. It comprises 7 metrics in 9 studies (22%); and

Table 5. Metrics to Measure the Developers' Work

#	Metrics	Interpretation	Reference	#	Metrics	Interpretation	Reference
1	% of Commits	Con	P22, P8	**28**	Contribution Start	Con, ImP	P34
2	% of Lines of Code	ImP, Con	P19, P35	**29**	Cost	Pro	P2
3	Authorship by Guilt	Con	P35	**30**	CQI	Qua	P14
4	CDI	Qua	P14	**31**	Developer Fragmentation	Con	P14, P22, P24
5	Change Code Documentation (CAD)	Con	P8, P11	**32**	Degree of Authorship (DOA)	Con	P29, P35
6	CK Metrics	Qua	P9, P38	**33**	Effort on commit	Pro, Con	P17
7	Close a Bug (BCL)	ImP	P9, P19, P8	**34**	Effort per Modification	Pro, Con	P25, P2
8	Close a Bug that is then Reopened (BCR)	ImP	P8	**35**	Expected Shortfall (ES)	Con, ImP	P37
9	Close a Lingering Thread (MCT)	Pro	P8	**36**	Expertise	ImP	P18, P31
10	Code Duplication	Qua	P11	**37**	Expertise Breadth of a Developer (EBD)	Con, ImP	P26
11	Collaboration (CodeChurn)	CoW, Con	P7	**38**	Expertise of a Developer (ED)	Con, ImP	P21, P26
12	Collaboration (Files)	CoW, Con	P14, P3, P10, P12, P30	**39**	First Reply to Thread (MRT)	ImP	P8
13	Collaboration (Interaction)	CoW, Beh	P21, P40	**40**	Hero	Con, ImP	P13
14	Comment on a Bug Report (BCR)	Con	P19, P8	**41**	Knowledge at Risk (KaR)	Con, ImP	P37
15	Commit Binary Files (CBF)	Con	P8	**42**	Knowledge Loss	Con, ImP	P32, P37
16	Commit Code that Closes a Bug (CCB)	Con	P39, P8	**43**	Last Committer "Takes All"	Con	P35
17	Commit Comment that Includes a Bug Report Num (CBN)	Con, ImP	P17, P19, P8	**44**	Link a Wiki Page from Documentation File (WLP)	Qua	P8
18	Commit Documentation Files (CDF)	Con	P39, P8	**45**	Martin Metrics	Qua	P38, P11
19	Commit Fixes Code Style (CSF)	Qua, Con	P39, P8	**46**	Mastery of Technologies	ImP, Con	P27, P33
20	Commit Multiples Files in a Single Commit (CMF)	Con	P19, P22, P28, P8	**47**	Monthly Effort	Pro	P20, P2
21	Commit New Source File or Directory (CNS)	Con	P8	**48**	MTBC	Pro	P34
22	Commit with Empty Commit Comment (CEC)	Con	P8	**49**	Number of Active Days	Con, ImP	P9, P12, P20, P19, P30, P34, P29, P39
23	Commitment	ImP, Beh	P7, P33, P15	**50**	Number of Code Churn	Con	P20, P34
24	Commits Versatility	Con	P34	**51**	Number of Commits	Con	P14
25	Complexity and Size	Qua	P9, P23, P38, P11	**52**	Number of Lines of Code (NLOC)	Con, CoW	P16, P19, P3, P9, P14, P21, P18, P22, P23, P39, P8, P40
26	Contribution Duration	Con, ImP	P34	**53**	Participate in a Flame War (MFW)	Beh	P8, P14
27	Contribution Factor	Con	P8	**54**	QCTE	Qua	P5, P4, P14, P35

Table 5. Metrics to Measure the Developers' Work (cont.)

#	Metrics	Interpretation	Reference	#	Metrics	Interpretation	Reference
55	QMood Metrics	Qua	P38	**60**	Start a New Wiki Page (WSP)	Con	P8
56	Report a Bug (BRP)	ImP	P8	**61**	Status	ImP	P22
57	Rework	Qua	P25, P33	**62**	Task Delivery	Pro	P24, P33, P36
58	Source Abandoned	Con, ImP	P32, P37	**63**	Truck Factor	Con, ImP	P13, P29, P35
59	Start a New Thread (MST)	Con	P8	**64**	Update a Wiki Page (WUP)	Con	P22

- **Behavior (Beh).** This group refers to the developers' behavior, such as focus, proactivity, communication skills, and interaction with the team, to identify members' engagement and commitment to the project (P7, P8, P15, P19, P22, P33). It comprises 5 metrics in 6 studies (14.6%).

For example, the Truck Factor metrics calculate the minimum number of developers who have to leave a project for it to be delayed (high probability). If this measurement returns a value of 1, it indicates that one developer has all the knowledge on the project. Therefore, project development can be delayed if this developer abandons it. Analogously, if this measurement returns a value of 3, it indicates that 3 developers have all the knowledge on the project. Thus, the project's development can be delayed if these developers abandon it. If only a single developer has the all knowledge on the project, the Hero metric indicates that this one is a "hero". Both metrics consider the developers' contribution towards devising the files, meaning their level of expertise in the project (Con, ImP).

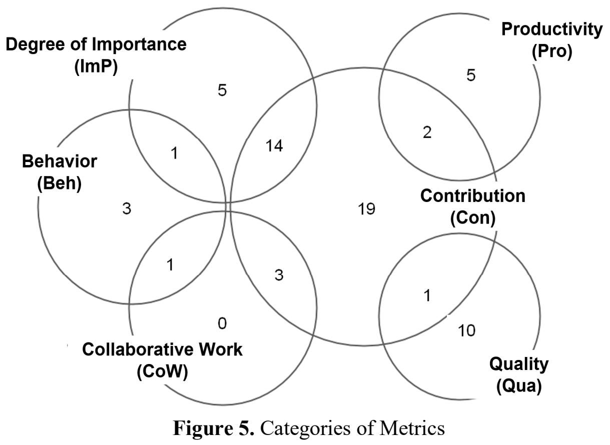

Figure 5. Categories of Metrics

To answer the primary research question

Q-P.2 - How are metrics applied by PMs to monitor the developers' work?

we considered (i) data sources used to obtain information on the developers' work (Table 6), (ii) the extraction method of this information for the application of metrics (Table 7), (iii) context in which the metrics were extracted (Figure 6), and (iv) presentation of the results for PMs to analyze (Table 8).

We identified 5 data sources: i) **Source Code Repositories** (supported by Version Control Systems) in 28 studies (68.3%); ii) **Bug Repositories** in 7 studies (17.1%); iii) **Activity Management Repositories** in 4 studies (9.8%); iv) **Source Code Files** in 5 studies (12.2%); and v) **People** (team members and/or PMs) in 7 studies (17.1%). We would like to point out that 8 studies used more than one data source (19.5%). The first four data sources provided impersonal data, and only the last one offered subjective data.

Table 6. Data Sources on the Developers' Work

Data Sources	References
Code Repositories	P3, P5, P8, P10, P9, P12, P13, P14, P16, P19, P20, P21, P22, P24, P23, P25, P26, P28, P29, P30, P31, P32, P34, P35, P37, P38, P39, P40
Bugs Repositories	P8, P9, P17, P19, P23, P25, P39
Activity Management Repositories	P8, P19, P22, P36
Source Code Files	P2, P4, P11, P17, P18
People	P1, P6, P7, P15, P27, P33, P41

Table 7. Strategies for Data Extraction and Metrics Application

Strategies	References
Automated Tools	P2, P3, P4, P5, P8, P9, P10, P12, P11, P13, P14, P16, P17, P18, P19, P20, P26, P28, P29, P30, P31, P35, P36, P39, P40
Manual Process	P21, P22, P23, P24, P25, P32, P34, P37, P38
Questionnaires, PMs' Intuition, Focal Groups, and Interviews	P1, P6, P7, P27, P33, P15, P41

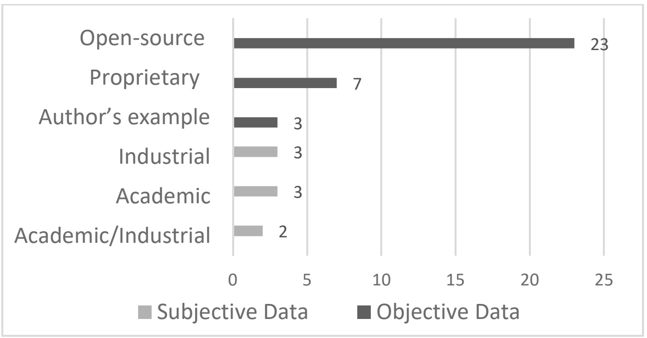

Figure 6: Extraction Context of Value of the Metrics

We found two strategies for data extraction and metrics application from impersonal data sources (Table 7): i) use of automated tools (25 studies - 61%); and ii) manual process (9 studies - 22%). We considered the impersonal data source to identify a strategy composed of questionnaires, PMs' intuition, focal groups, and interviews (7 studies - 17.1%).

Table 8. Strategies to Present Information

Presentation Form	References
Textual	P1, P4, P13, P14, P16, P19, P18, P22, P25, P26, P27, P29, P30, P31, P34, P35, P36, P39
Graphical	P2, P9, P10, P13, P11, P14, P17, P16, P18, P19, P20, P22, P25, P26, P31, P34, P38, P40
Visualization Techniques	P3, P5, P7, P9, P10, P12, P16, P28, P30, P39

Additionally, we analyzed the data source used to identify the metrics application context (Figure 6). We analyzed the type of software in which the extraction of impersonal data used metrics (33 studies): i) **proprietary software** (7 studies - 21.2%); ii) **open-source software** (23 studies - 69.7%); and iii) **author's example** (3 studies - 9.1%). For the impersonal data extraction, we analyzed the contexts in which interviews, questionnaires, and focus groups were held (8 studies): i) **academic** (3 studies - 37.5%); ii) **industrial** (3 studies - 37.5%); and iii) **industrial and academic** (2 studies - 25%).

For the extraction of impersonal data in an open-source context, the metrics were used with a frequency (median) of three software per study (P3, P4, P5, P8, P9, P10, P12, P13, P14, P16, P17, P18, P19, P20, P26, P29, P30, P31, P34, P35, P36, P38, P39 - 23 studies). The most widely used open-source software systems for data extraction were Eclipse (P4, P17, P19, P31 - 4 studies), JEdit (P5, P9, P13, P30 - 4 studies), Apache (P26, P31, P36 - 3 studies), and OpenStack (P20, P31 - 2 studies). In the proprietary software context, the studies used metrics in just one system (P11, P16, P21, P23, P25, P32, P40 - 7 studies). In the context the author used as an example, they used examples to show the metrics value extraction (P22, P24, P28 - 3 studies). In order to extract subjective data in the academic context, undergraduate students in computer science and business informatics were interviewed (P1, P15, P41 - 3 studies). In the industry context, interviews with product engineers and project managers were carried out (average = 12 people/study) (P2, P6, P7, P27, P33 - 5 studies).

From the context of classification used for data extraction, we identified that the studies analyzed the same metric in different contexts. Therefore, the metrics' applicability to measure the developers' work does not depend on a specific context. It is sufficient to use metrics on software (proprietary or open-source) and developers for the impersonal and subjective data extraction, respectively.

Besides, we identified three strategies to submit information to PMs (32 studies - 78%) (Table 8): i) **Textual** (18 studies - 56.3%), ii) **Graphical** (18 studies - 56.3%), and iii) **Visualization Techniques** (10 studies - 31.3%). We point out 13 studies combining two or more strategies (40.6%).

To answer the primary research question

Q-P.3 - How do metrics concerning the developers' work support project management, especially with regard to risk management and people management?

we identified the project management activities supported by the information obtained by measuring the developers' work. It is worth noticing that we defined these activities using the researchers' interpretation. We analyzed the objectives and findings of the studies regarding the background presented (Section 2). For example, Lima and Elias 2019 (Lima and Elias, 2019) proposed a systematic approach to assign people to a specific activity according to their personality and skills. We used this concept in order to categorize the activities. After extracting the data, we merged some categories according to the similarity of their names and definition. In total, we found 7 activities that support project management (Table 9), and we highlight the fact that 16 studies (48.5%) treated two or more activities:

- **Identify the skills and the profile of developers** (25 studies - 61%);
- **Plan improvements to code quality** (4 studies - 9.8%);
- **Improve team performance** (12 studies - 29.3%);
- **Estimate project costs and deadlines and identify anomalies in developer performance** (15 studies - 36.6%);
- **Understand and control knowledge distribution** (9 studies - 22%);
- **Adjust pay** (2 studies - 4.9%); and
- **Identify the need for investment (training or equipment)** (5 studies - 12.2%).

6 Discussion

In this section, we discussed the results obtained with the SMS and presented the four PMs' opinions. We collected their opinions to understand to what extent the metrics collected could be used to analyze the developers' work. The PMs' opinions were collected using the interview described in Appendix B.

There is a wide variety of existing metrics to measure the developers' work (64 metrics). We found two most commonly used metrics: NLOC (number of lines of code) (12 studies) and number of commits (9 studies). Table 10 presents 45 metrics created from those metrics above. There are 17 metrics unrelated to NLOC and the number of commits (Table 11) and associated with bugs (2 metrics), documentation (6 metrics), behavior (2 metrics), and quality (7 metrics) activities.

Table 9. Activities Supported by Information about the Developers' Work

Activities	References
Identify the skills and the profile of developers	P24, P1, P3, P5, P7, P8, P9, P10, P11, P12, P14, P15, P20, P21, P22, P23, P25, P27, P30, P31, P33, P34, P36, P40, P41
Plan improvements to code quality	P14, P16, P22, P24
Improve team performance	P1, P7, P12, P15, P16, P18, P19, P22, P23, P32, P33, P37
Estimate project costs and deadlines and identify anomalies in developer performance	P1, P2, P6, P8, P9, P11, P12, P14, P17, P19, P18, P22, P28, P34, P33
Understand and control knowledge distribution	P8, P13, P23, P26, P29, P32, P33, P35, P37
Adjust pay	P19, P23
Identify the need for investment (training or equipment)	P8, P15, P17, P24, P28

Table 10. Metrics Created from NLOC and Number of Commits

#	Metrics	#	Metrics
1	% of Commits	20	Contribution Duration
2	% of Lines of Code	21	Contribution Factor
3	Authorship by Guilt	22	Contribution Start
4	Close a Bug (BCL)	23	Cost
5	Close a Bug that is then Reopened (BCR)	24	Developer Fragmentation
6	Close a Lingering Thread (MCT)	25	Degree of Authorship (DOA)
7	Code Duplication	26	Effort on Commit
8	Collaboration (CodeChurn)	27	Effort per Modification
9	Collaboration (Files)	28	Expected Shortfall (ES)
10	Comment on a Bug Report (BCR)	29	Expertise Breadth of a Developer (EBD)
11	Commit Binary Files (CBF)	30	Expertise of a Developer (ED)
12	Commit Code that Closes a Bug (CCB)	31	Expertise
13	Commit Comment that Includes a Bug Report Number (CBN)	32	Hero
14	Commit Documentation Files (CDF)	33	Knowledge at Risk (KaR)
15	Commit Fixes Code Style (CSF)	34	Knowledge Loss
16	Commit Multiple Files in a Single Commit (CMF)	35	Last Committer "Takes All"
17	Commit New Source File or Directory (CNS)	36	Mastery of Technologies
18	Commit with Empty Commit Comment (CEC)	37	Monthly Effort
19	Commits Versatility	38	MTBC

Table 10. Metrics Created from NLOC and Number of Commits (cont.)

#	Metrics	#	Metrics
39	Number of Active Days	43	Status
40	Number of Code Churn	44	Task Delivery
41	Participate in a Flame War (MFW)	45	Truck Factor
42	Source Abandoned		

Table 11. Unrelated Metrics to NLOC and Number of Commits

#	Purpose	Metrics
1	Bugs	Rework
2		Report a Bug (BRP)
1	Documentation	Change Code Documentation (CAD)
2		First Reply to Thread (MRT)
3		Link a Wiki Page from Documentation File (WLP)
4		Start a New Thread (MST)
5		Start a New Wiki Page (WSP)
6		Update a wiki Page (WUP)
1	Behavior	Commitment
2		Collaboration (Interaction)
1	Quality	CDI
2		CK Metrics
3		CQI
4		Martin Metrics
5		QCTE
6		QMood Metrics
7		Size

Despite the variety of metrics, most of them quantify the developers' work regarding their contributions to the software artifacts development. Hence, 34 metrics form the Contribution group, which shares metrics with the other groups created, except the Behavior Group, because it considers aspects such as focus and commitment, not the work done by the developers.

PMs could choose any metrics from the 64 metrics identified in SMS as being essential to measure the developer's work. As a result, they mentioned 29 metrics, of which two or more PMs cited the same 9 metrics. Three PMs chose 3 metrics (Task Delivery, Number of Commits, and Rework) showing that the performance on task delivery, frequency of contribution to the artifacts and solution generation do not need reworking as the most relevant inputs to measure the developers' work. Another interesting point is the perception of the two PMs who work in the same company (PM1 and PM4). Although they have the same amount of experience in Company A, they have different opinions with regard to measuring the developers' work. While PM1 highlighted only 1 metric, PM4 highlighted 24 metrics. This result shows the difficulty in finding a consensus regarding how to measure the developers' work. None of the PMs suggested using a metric beyond the 64 metrics presented in Table 5. Nevertheless, PM1 mentioned code smells as a possible metric, but he/she did not justify its use. Maybe, the "number of code smells" can be one metric used to verify the quality of solutions implemented by the developer, but we need to define how to use it. Besides, PM2 said, "I use the Task Delivery, Status, and Contribution Duration metrics using different names (Throughput, Lead Time, and Cycle Time, respectively)".

One interesting factor is that the selected studies used more than one metric for measuring the developers' work. This may occur due to task complexity, requiring a lot of data to analyze the team members' performance in the fairest way possible. The proposed organization of the metrics into groups provides an overview of the "types of information" used by PMs. However, no studies covered metrics from all groups.

With regard to the artifact used to extract the value of the metrics, we found the source code repositories (managed by Version Control Systems) to be the most frequently used to collect information about the developers' work. The information collected includes mainly the execution of commits and addition/modification/removal of lines of code. As previously mentioned, they indicate the degree of contribution to the project and are bases for most of the metrics listed. For example, we can trace the activity performed with the commit message if this activity consisted of a bug fixing. Another interesting point about NLOC and the number of commits is to apply filters, which can be: i) **Information granularity levels** (*e.g.*, the number of commits performed by one developer in a directory, file or line of code); and ii) **Time intervals** (*e.g.*, how much time the developer takes adding lines of code to the repository or how much time the developer contributed to the project in recent months).

Another example of existent information in the source code repositories is the degree of collaboration among developers. The variability of possibilities offered by the data obtained from the source code repositories is numerous and diverse, in addition to providing support for the two most representative metrics (number of commits and NLOC). It can justify the code repositories to be the data source most often used in the studies found in SMS. The source code files are other data sources but limited to questions of complexity and quality code. The source code file analysis in an integrated development environment (*e.g.*, Eclipse IDE) overcomes that limitation, which consists of enhancing it with plugins and enriching information. For example, we can determine how long one developer has kept a source code file open or which developer changed the source code.

Using bug repositories helps to identify the developers' work regarding bug detection, correction, or generation in software, providing information on when the developer contributed to correct bugs or need of reworking. Task repositories allow managing of bug activities, which can be supported by tools, such as spreadsheets or specialized tools. By analyzing the records of tasks performed by the developers, it is possible to identify information such as the history of tasks delivered by the developer, including extra data (*e.g.*, duration), and types of tasks. This information can be extracted manually or automatically (through integration with the task repositories).

People's opinions were collected through feedback from team members and are based on the PMs' intuition and knowledge about the developers, thus producing more subjective data about their work. Possible information includes type and frequency of activities, technical and personal skills, satisfaction, motivation, behavior, and personality. Questionnaires, interviews, or focus groups are used to extract such information.

Interestingly, source code repositories used with other data sources are trivial and are a valuable data source to obtain information about the developers' work. Again, the complexity of quantifying developers' work explains this use. Such complexity can be the reason for using automated tools for data extraction and metrics application and is the most commonly used strategy to obtain the metrics on the developers' work (28 studies - 68.3%). Automated tools include web systems, software executed via command terminals, and plugins for integrated development environments. The web systems offer more value to PMs among the automated tools because the other tools have no graphical interface and are limited, and any devices and operating systems (using a web browser) can access web systems.

Another aspect observed in SMS, motivated by the complexity of measuring the developer's work, is the balance between the three ways of presenting the information measured to PMs (textual - 56.3%, graphical - 56.3%, and visualization techniques - 31.3%). The studies used text to show uncomplicated information, graphical for group information, and visualization techniques when the amount of information displayed was higher. Several studies (40.6%) used a combination of these presentation forms.

We also noticed that 7 activities inherent to PMs can be supported by measuring the developers' work. The activities can be related to the definitions described in PMBoK for People Management, Risk Management, Project Quality Management, and Resource Management. In "Identifying the skills and profile of the developer" activity, the PM chooses the members for a project team considering the skills and professional profile required to achieve the project goals. This activity supports the task allocation, considering the most appropriate person to perform a function (People management). Risk management can be supported, for example, in the following way: the PM can consider that the people available for the project do not have sufficient project technology skills (risk: not obtaining excellent project performance). Therefore, close monitoring of the team's work is necessary, as well as hiring a consultancy firm.

In the interview, the PMs pointed out the metrics they consider useful to carry out the "Identify the skills and the profile of developers" activity. Thus, they identified 21 metrics among the 64 metrics found in SMS. The most frequently mentioned metric (three PMs) was Commitment. This metric provides information on the developer's behavior. Other metrics mentioned by more than one PM were Collaboration (Interaction), Mastery of Technologies, Contribution Factor, Complexity and Size (5 metrics). The

PMs' vision is to measure how committed the developer is to the project and the team, how much the developer collaborates with colleagues, and their mastery of project technologies, enabling them to contribute to the implementation of complex solutions for the software. Besides, more than one PM considered other relevant metrics. Such metrics provide information about the complexity of tasks performed by the developer, the contribution factor in software artifacts, the technical capacity in design technologies, and performance in collaboration with colleagues.

In the "Planning Improvements to Code Quality" activity, PMs analyze the quality of the solutions delivered by the developers. The answers must meet the expectations of those involved with the project (including team, PMs and customers) and should be good enough to avoid rework. Quality is a factor related to project management, and there is an area of knowledge in PMBoK called Project Quality Management for this factor.

In the interview, PMs pointed out the metrics they consider useful to carry out the "Planning Improvements to Code Quality" activity. Thus, they identified 17 metrics among the 64 metrics found in SMS. Two or more PMs mentioned the same 2 metrics (Rework and Collaboration (interaction)). The PMs' view of this activity is significantly divergent. In the opinion of PM1, documentation is the main point, considering the writing software-specific documents and the documentation made from commits in code repositories. For PM2, the solution-generated quality and bugs reported for lines of code that the developers created are interesting for quality checking. PM3 considers that developer's experience and how much knowledge he/she shares with other team members are the main way to ensure that code quality is satisfactory. For PM4, object-oriented code quality metrics, the quality of the implemented solution and the developer's ability to work in various code parts provide inputs to observe source code quality. In general, metrics are used for documentation, implementation quality, and generated rework.

In "Improving Team Performance" activity, there is a dependency on measuring the developers' work, because measuring the team's performance (to determine the current state) is necessary in order to plan and apply actions to leverage performance (to promote improvement) and to remeasure it (to the new state). This activity is essential for people management, such as defining teams, assigning roles, communicating, and organizing work, thus contributing to performance improvement.

In the interview, PMs pointed out the metrics they consider useful to carry out the "Improving Team Performance" activity. Thus, they identified 15 metrics among the 64 metrics found in SMS. Out of these, two or more PMs mentioned the same 6 metrics (Collaboration (Files), Collaboration (Interaction), Mastery of Technologies, Expertise, Collaboration (CodeChurn), and Closing a Bug that is then Reopened (BCR)). PMs share a similar view on performance: the developer with the technical knowledge, experience, and participation in various parts of source code gets the best performance. Besides, performance is analyzed by looking at not just one developer, but at the entire team. Hence, one developer cannot produce new solutions, but collaborate with other developers for the team to deliver an answer as soon as possible. PM2 and PM3 also highlighted the quality of the implemented solution that the developer performs better, as corrections or refactoring are necessary when the developer provides an error-free task and satisfactory quality. In the same way as the previous activity (related to quality), PMs considered metrics related to reworking in order to understand the team's performance.

In the "Estimating Project Costs and Deadlines and Identifying Anomalies in Developer Performance" activity, there is a relationship with people management because the project budget and cost are anticipated when PMs record historical information on the developers' work. Besides, recent history can help to identify when one developer is performing differently than expected (better or worse), which can be a consequence of a change in the developer's motivation or his/her interpersonal relationships with the team.

In the interview, PMs pointed out the metrics they consider useful to carry out the "Estimating Project Costs and Deadlines and Identifying Anomalies in Developer Performance" activity. Thus, they identified 24 metrics among the 64 metrics found in SMS. Two or more PMs mentioned the same 6 metrics (Cost, Effort per Modification, Rework, Commitment, Task Delivery, and Contribution Factor). The four PMs interviewed considered the technical ability and delivery history of the team allocated to the project to be essential in order to estimate costs and deadlines. Additionally, they said that the developers' knowledge and contributions to the project's source code should be taken into account in the estimates. PM1 pointed out that performance anomalies are usually caused by the emergence of unplanned and highly complex demands, leading to high development costs.

In the "Understanding and Controlling Knowledge Distribution" activity, there is support for risk management because the departure of one developer who has most of the knowledge about the source code of a project can present high risks. Therefore, PMs should monitor knowledge distribution, act towards leveling the knowledge of the team, and prevent essential people from leaving the project earlier than expected.

In the interview, PMs pointed out the metrics they consider useful to carry out the "Understanding and Controlling Knowledge Distribution" activity. Thus, they identified 17 metrics among the 64 metrics found in SMS. Out of the 64, two or more PMs mentioned the same 6 metrics (Commit Documentation Files (CDF), Collaboration (Files), Collaboration (Interaction), Collaboration (CodeChurn), Change Code Documentation (CAD), and Degree of Authorship (DOA)). In the PMs' opinion, it is possible to identify a developer's knowledge of the system by observing the contributions in writing their documentation and code. PM3 also considers commitment, time working on the project, and all activities registered in the version control system.

Another interesting point is to measure how much the developer shares your knowledge. PM2, PM3, and PM4 highlighted this information. Among the studies found in SMS, one study addressed knowledge distribution with greater emphasis (Ferreira et al. 2017). The authors calculated the DOA metric for all files in the code repository in order to identify the developers with the most knowledge of the project. This calculation generated another metric (Truck Factor metric). However, when choosing the DOA metric instead of the Truck Factor metric, PMs preferred to look at individual files instead of the entire set of files in the repository, identifying developers' knowledge in parts of source code.

In the "Pay Adjustments" activity, the work done by the development team is recognized, thus keeping professionals motivated and satisfied. This activity is critical when managing people. In the "Identifying the Need for Investment" activity, there is acquisition of equipment and training in order to meet project needs.

In the interview, PMs pointed out the metrics they consider useful to carry out the "Pay Adjustments" activity. Thus, they identified 30 metrics among the 64 metrics found in SMS. Out of these 30, two or more PMs mentioned the same 12 metrics (Commitment, Expertise, Rework, Collaboration (Files), Mastery of Technologies, Task Delivery, Monthly Effort, Contribution Factor, Number of Active Days, Cost, Contribution Start, and Contribution Duration). With regard to other activities, this one received the most significant number of different metrics, and the rate of metrics chosen by more than one PM was higher. This variety of metrics may be due to the activity's sensitivity and the concern for making compensation adjustments in the most adequate way. In general, according to the PMs, the developers' commitment and technical ability, the quality of the solutions generated and the time spent on the project are primary information.

In the interview, PMs pointed out the metrics they consider useful to carry out the "Identifying the Need for Investment" activity. Thus, they identified 23 metrics among the 64 metrics found in SMS. Two or more PMs mentioned the same 3 metrics (Close a Bug that is then Reopened (BCR), Mastery of Technologies, and Effort on Commit). For PMs, it is necessary to understand if the developers are technically competent to perform their activities. Hence, the selected metrics provide the level of effort to accomplish tasks, the mastery of project technologies, and the number of errors and rework for solutions delivered by developers.

In Table 12, we presented the metrics that PMs cited ten or more times. They considered them relevant, and their opinion provides some (few) professionals' views on the metrics found in the literature. Thus, these results cannot be generalized for the entire industrial context.

Table 12. Metrics most selected by PMs

#	Metric	Number of Citations	Mentioned by
1	Collaboration (Interaction)	15	PM2: 1, 3, 5 PM3: 1, 2, 3, 4, 5, 6, 7, General PM4: 1, 3, 5, General
2	Rework	14	PM2: 1, 2, 3, 4, 5, 6, 7, General PM3: 6, General PM4: 2, 4, 6, General
3	Collaboration (Files)	12	PM2: 3, 5, 6 PM3: 3, 4, 5, 6, 7 PM4: 1, 3, 5, General
4	Expertise	12	PM1: 3 PM2: 6 PM3: 1, 2, 3, 5, 6, 7, General PM4: 4, 6, General
5	Mastery of Technologies	12	PM1: 3 PM2: 1, 3, 6, 7 PM3: 3, 5, General PM4: 1, 6, 7, General
6	Commitment	11	PM1: 6 PM2: 1, 4, 6 PM3: 1, 3, 4, 5, 6, General PM4: 1
7	Cost	11	PM2: 4 PM3: 1, 2, 3, 4, 6, 7, General PM4: 4, 6, General

1. Identify the skills and the profile of developers; 2. Plan improvements to code quality; 3. Improve team performance; 4. Estimate project costs and deadlines and identify anomalies in developer performance; 5. Understand and control knowledge distribution; 6. Adjust pay; 7. Identify the need for investment (training or equipment); General. Considers relevant to project management (unspecified activity).

7 Threats to Validity

Internal Validity. Refers to the effects of the treatments over the variables due to uncontrolled factors in the environment (Wohlin et al. 2012). Limitations of the search string and digital libraries can lead to an incomplete selection of studies. We selected five search engines and adapted the search string to achieve our goal. However, other digital libraries and keywords could be added to the search string. Another possible threat is the researchers' bias in selecting the studies, and answering the research questions (*e.g.*, identification and grouping of metrics, interpretation of project management activities). To mitigate this, Researchers A and B discussed the results while generating the groups presented. These researchers were monitored by Researcher C. Finally, Researcher D (most experienced researcher) evaluated the work performed and suggested improvements to ensure the impartiality and quality of this study. Moreover, we carefully defined and reported the search string, digital libraries chosen, and the inclusion and exclusion criteria to ensure SMS replicability.

External Validity. Relates to whether the results can be generalized outside the experimental setting (Wohlin et al. 2012). One threat to external validity is about our selecting representative studies. With regard to the amount of information collected, we argue that the selected studies are representative. However, we only considered studies from the formal literature, which could be extended by considering the gray literature. Our findings are focused on evaluating the developers' work. Currently, we have no intention to generalize our results beyond this field.

Construct Validity. Represents the measurement of the concepts of cause and effect in the experiment through dependent and independent variables (Wohlin *et al.* 2012). To ensure that SMS was impartial, comprehensive, and of high quality, four researchers took part in the definition and execution of the research protocol. The protocol used to select the studies was validated using a control group (7 studies). However, we addressed 6 studies from this group and added the other study manually. It was one consequence of our decision to limit results by applying keywords from Parts 2, 3, 4, and 5 just in the title, abstract, and keyword of the studies. With regard to data extraction, we defined a classification scheme. However, it was a manual process, and we cannot claim that it was carried out mistake-free. The data extraction process required an understanding of the subject to infer the non-explicit data, which made this process exhaustive and complicated.

Conclusion Validity. Refers to the extension of the conclusions about the relationship between the treatments and the outcomes (Wohlin *et al.* 2012). We followed a systematic approach for conducting SMS and described all procedures to ensure this study's replicability.

8 Final Remarks

Project managers (PMs) are professionals whose task is to successfully lead software projects. Therefore, project management practices are applied. Aiming to support the PM, several studies in the literature have proposed strategies to measure the developers' work. In this article, an exploratory study was carried out by using the Systematic Mapping Study (SMS) and interviewing PMs to organize concepts related to this topic.

In this context, we answered three primary research questions, specific to our study field (Q-P.1 - What metrics are used by PMs to measure the developers' work?, Q-P.2 - How are metrics applied by PMs to monitor the developers' work?, and Q-P.3 - How does information about the developers' work support project management?). Additionally, we answered three secondary research questions, which are common to SMS studies (Q-S.1 - What type of solution is often proposed for studies in this area?, Q-S.2 - What type of research methodology is often used for studies in this area?, and Q-S.3 - How is the proposed solution related to the research methodology in the included studies?). The responses were based on an analysis of 41 studies found using SMS and the opinions of four PMs opinions.

Our contributions are: i) identification of the studies' maturity, ii) identification of solution proposals to investigate the study theme, iii) identification of 64 metrics, iv) organization of the metrics into 6 groups, v) identification of 4 data sources to obtain information, vi) identification of the data extraction context and metrics application, vii) identification of 7 activities for which the PM is responsible (most of these activities are related to risk management and people management), supported by a measurement of the developers' work, viii) the opinion of four PMs on the usefulness of the 64 metrics; ix) the opinion of four PMs on how the 64 metrics relate to 7 activities under their responsibility, and x) characterization of the aspects to explore the subject, indicating themes for possible new studies in the area of Software Engineering.

The suggestions for future work include: i) verification of the validity of metrics found by collecting the most significant number of PM opinions, allowing a quantitative analysis, ii) assessment of the validity of the metrics found by conducting a field study, iii) evaluation of approaches to measure the developers' work considering the industrial, and proprietary software context, iv) creation of new approaches (or advances in existing approaches) to consider diversified metrics that provide information about work quality, contribution by the developer, collaboration, level of importance to the project, productivity, and personal behavior, and v) combination of three forms of presenting information (textual, graphical, and visualization techniques).

References

B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," 2007.

B. Kitchenham, "Procedures for Performing Systematic Reviews," Keele, UK, Keele Univ., vol. 33, pp. 1-26, 2004.

B. W. Boehm, "Software Risk Management: Principles and Practices," IEEE Softw., vol. 8, no. 1, pp. 32–41, 1991.

C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. Experimentation in Software Engineering. Springer Science & Business Media, New York, NY. 2012.

I. Bouchrika, "Top Computer Science Conferences", Guide2Research, 2020, available at <http://www.guide2research.com/topconf/>, last access May, 9th, 2020.

I. Sommerville, Engenharia de Software, 10th ed. Pearson Universidades, 2019.

J. A. Lima and G. Elias, "Selection and Allocation of People based on Technical and Personality Profiles for Software Development Projects," XLV Latin American Computing Conference (CLEI), Panama, 2019, pp. 1-10, doi: 10.1109/CLEI47609.2019.235052.

J. Feiner and K. Andrews, "RepoVis: Visual Overviews and Full-Text Search in Software Repositories," in Working Conference on Software Visualization, 2018, pp. 1–11.

J. J. Ahonen, P. Savolainen, H. Merikoski, and J. Nevalainen, "Reported project management effort, project size, and contract type". Journal of Systems and Software 109, 2015, pp. 205–213.

J. Menezes, C. Gusmão, and H. Moura, "Risk Factors in Software Development Projects: A Systematic Literature Review", in Software Qual J 27, 2019, pp 1149–1174.

K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, "Systematic Mapping Studies in Software Engineering", in: Presented at the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008.

K. Tuma, Ç. Gül, and S. Riccardo. "Threat analysis of software systems: A systematic literature review", Journal of Systems and Software 144, 2018, pp. 275-294.

M. Ferreira, M. T. Valente, and K. Ferreira, "A Comparison of Three Algorithms for Computing Truck Factors," IEEE/ACM 25th International Conference on Program Comprehension, 2017, pp. 207–217.

N. Bin Ali and K. Petersen, "Evaluating Strategies for Study Selection in Systematic literature studies," In: ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2014, p. 45.

N. Wieringa, N. M. R. Maiden and C. Rolland. Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion. Requirements engineering, v. 11, n. 1, p. 102-107, 2006.

P. R. de Bassi, G. M. P. Wanderley, P. H. Banali, and E. C. Paraiso, "Measuring Developers' Contribution in Source Code using Quality Metrics," in IEEE International Conference on Computer Supported Cooperative Work in Design, 2018, pp. 39–44.

PMI, Guide to the Project Management Body of Knowledge (PMBOK® Guide), 6th ed., 2017.

R. Latorre and S. Javier, "Measuring social networks when forming information system project teams", Journal of Systems and Software 134, 2017, pp. 304-323.

R. Martin, "OO Design Quality Metrics - An Analysis of Dependencies," in Workshop Pragmatic and Theoretical Directions in Object-Oriented Software Metrics, 1994.

S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Trans. Softw. Eng., vol. 20, no. 6, pp. 476–493, 1994.

T. Ambreen, N. Ikram, M. Usman, and Niazi, M. "Empirical Research in Requirements Engineering: Trends and Opportunities". Requirements Eng 23, 63–95 (2018).

V. F. de Souza, A. L'Erario, and J. A. Fabri, "Model for Monitoring and Control of Software Production in Distributed Projects". In: Iberian Conference on Information Systems and Technologies, 2015, pp. 1–6.

V. Garousi; Y. Amannejad; A. B. Can. "Software Test-Code Engineering: A Systematic Mapping". Information and Software Technology, v. 58, p. 123-147, 2015.

W. Zuser and T. Grechenig, "Reflecting Skills and Personality Internally as Means for Team Performance Improvement." In: Conference on Software Engineering Education and Training, 2003, pp. 234–241.

Appendix A

Table A1. Resultant Studies from SMS

#	Title	Authors	Year	Repository	Score
P1	Reflecting Skills and Personality Internally as Means for Team Performance Improvement	Zuser, W. Grechenig, T.	2003	IEEE	3.5
P2	Continuous productivity assessment and effort prediction based on Bayesian analysis	Yun, S. Simmons D.	2004	Ei Compendex	5.0
P3	Visualization of CVS Repository Information	Xie, X. Poshyvanyk, D. Marcus, A.	2006	IEEE	4.5

Table A1. Resultant Studies from SMS (cont.)

#	Title	Authors	Year	Repository	Score
P4	A 3-Dimensional Relevance Model for Collaborative Software Engineering	Omoronyia, I. Ferguson, J. Roper, M. Wood, M.	2007	IEEE	3.5
P5	A Visualization for Software Project Awareness and Evolution	Ripley, R. Sarma, A. van der Hoek, A.	2007	IEEE	5.0
P6	Evaluating Software Project Portfolio Risks	Costa, H. Barros, M. Travassos, G.	2007	Ei Compendex	4.0
P7	Development of a Project Level Performance Measurement Model for Improving Collaborative Design Team Work	Yin, Y. Qin, S. Holland, R.	2008	IEEE	4.5
P8	Measuring Developer Contribution from Software Repository Data	Gousios, G. Kalliamvakou, E. Spinellis, D.	2008	ACM	4.0
P9	Mining Individual Performance Indicators in Collaborative Development Using Software Repositories	Zhang, S. Wang, Y. Xiao, J.	2008	IEEE	4.0
P10	SVNNAT: Measuring Collaboration in Software Development Networks	Schwind, M. Wegmann, C.	2008	IEEE	3.5
P11	Case Study: Visual Analytics in Software Product Assessments	Telea, A. Voinea, L.	2009	Ei Compendex	4.5
P12	Using Transflow to Analyze Open-Source Developers' Evolution	Costa, J. Santana Jr., F. de Souza, C.	2009	Scopus	4.5
P13	Are Heroes Common in FLOSS Projects?	Ricca, F. Marchetto, A.	2010	ACM	4.0
P14	PIVoT: Project Insights and Visualization Toolkit	Sharma, V. Kaulgud, V.	2012	IEEE	5.0
P15	Effect of Personality Type on Structured Tool Comprehension Performance	Gorla, N. Chiravuri, A. Meso P.	2013	Springer	4.0
P16	Extracting, Identifying and Visualisation of the Content, Users and Authors in Software Projects	Polášek, I. Uhlár, M.	2013	Scopus	5.0
P17	Towards Understanding How Developers Spend Their Effort During Maintenance Activities	Soh, Z. Khomh, F. Guéhéneuc, Y. Antoniol, G.	2013	IEEE	5.0
P18	A Machine Learning Technique for Predicting the Productivity of Practitioners from Individually Developed Software Projects	Lopez-Martin, C. Chavoya, A. Meda-Campana, M.	2014	IEEE	4.5
P19	Determining Developers' Expertise and Role: A Graph Hierarchy-Based Approach	Bhattacharya, P. Neamtiu, I. Faloutsos, M.	2014	IEEE	5.0
P20	Estimating Development Effort in Free/Open-source Software Projects by Mining Software Repositories: A Case Study of OpenStack	Robles, G. González-Barahona, J. M. Cervigón, C. Capiluppi, A. Izquierdo-Cortázar, D.	2014	ACM	5.0
P21	Extracting New Metrics from Version Control System for the Comparison of Software Developers	Moura, M. Nascimento, H. Rosa, T.	2014	IEEE	4.5
P22	Influence of Social and Technical Factors for Evaluating Contribution in GitHub	Tsay, J. Dabbish, L. Herbsleb, J.	2014	ACM	5.0

Table A1. Resultant Studies from SMS (cont.)

#	Title	Authors	Year	Repository	Score
P23	Assessing Developer Contribution with Repository Mining-Based Metrics	Lima, J. Treude, C. Filho, F. Kulesza, U.	2015	IEEE	4.0
24	Contributor's Performance, Participation Intentions, Its Influencers and Project Performance	Rastogi, A.	2015	IEEE	4.0
P25	Identifying Wasted Effort in the Field Via Developer Interaction Data	Balogh, G. Antal, G. Beszedes, A. Vidacs, L. Gyimothy, L. Vegh, T. Zoltan A.	2015	IEEE	4.5
P26	Niche vs. Breadth: Calculating Expertise over Time through a Fine-Grained Analysis	da Silva, J. Clua, E. Murta, L. Sarma, A.	2015	IEEE	5.0
P27	Proposal for a Quantitative Skill Risk Evaluation Method Using Fault Tree Analysis	Liu, G. Yokoyama, S.	2015	IEEE	4.0
P28	TeamWATCH Demonstration: A Web-based 3D Software Source Code Visualization for Education	Gao, M. Liu, C.	2015	Scopus	4.5
P29	A Comparative Study of Algorithms for Estimating Truck Factor	Ferreira, M. Avelino, G. Valente, M. Ferreira, K.	2016	IEEE	5.0
P30	Knowledge Discovery in Software Teams by Means of Evolutionary Visual Software Analytics	González-Torres, A. García-Peñalvo, F. Therón-Sánchez, R. Colomo-Palacios, R.	2016	Scopus	5.0
P31	Open-source Resume (OSR): A Visualization Tool for Presenting OSS Biographies of Developers	Jaruchotrattanasakul, T. Yang, X. Makihara, E. Fujiwara, K. Iida, H.	2016	IEEE	5.0
P32	Quantifying and Mitigating Turnover-Induced Knowledge Loss: Case Studies of Chrome and a project at Avaya	Rigby, P. Zhu, Y. Donadelli, S. Mockus, A. Rigb, P. Zhu, Y. Donadell, S. Mockus, A.	2016	Scopus	5.0
P33	Software Project Managers' Perceptions of Productivity Factors: Findings from a Qualitative Study	Oliveira, E. Conte, T. Cristo, M. Mendes, E.	2016	ACM	4.5
P34	Using Temporal and Semantic Developer-Level Information to Predict Maintenance Activity Profiles	Levin, S. Yehudai, A.	2016	IEEE	5.0
P35	A Comparison of Three Algorithms for Computing Truck Factors	Ferreira, M. Valente, M. Ferreira, K.	2017	IEEE	5.0
P36	Collabcrew - An Intelligent Tool for Dynamic Task Allocation within a Software Development Team	Samath, S. Udalagama, D. Kurukulasooriya, H. Premarathne, D. Thelijjagoda, S.	2017	IEEE	4.5

Table A1. Resultant Studies from SMS (cont.)

#	Title	Authors	Year	Repository	Score
P37	Revisiting Turnover-Induced Knowledge Loss in Software Projects	Nassif, M. Robillard, M. P.	2017	Scopus	3.0
P38	Measuring Developers' Contribution in Source Code using Quality Metrics	de Bassi, P. Wanderley, G. Banali, P. Paraiso, E.	2018	IEEE	3.5
P39	RepoVis: Visual Overviews and Full-Text Search in Software Repositories	Feiner, J. Andrews, K.	2018	IEEE	3.5
P40	git2net - Mining Time-Stamped Co-Editing Networks from Large git Repositories	Gote, C. Scholtes, I. Schweitzer, F.	2019	IEEE	4.5
P41	Selecting Project Team Members through MBTI Method: An Investigation with Homophily and Behavioural Analysis	Kollipara, P. Regalla, L. Ghosh, G. Kasturi, N.	2019	IEEE	3.0

Appendix B

1. Interview with PMs

We collected the opinion of professionals who work with project management in the software industry and used a structured script containing the following items for conducting the interview[1]:

- Description (company, level of education, experience, and number of projects managed);
- Opinion on which of the 64 metrics (Table 5) is necessary to measure the developers' work;
- Opinion on which of the 64 metrics (Table 5) are useful to support the performance of the 7 activities listed in Table 9; and
- Suggestion for other metrics.

We interviewed four PMs from three private companies with different characteristics. By interviewing these PMs, information from professionals in different contexts and projects was collected. The company's characteristics are:

- Company A is a software factory in the Brazilian market and has approximately 70 employees;
- Company B is a software factory operating in Brazil's education area, with around 150 employees; and
- Company C is an enterprise software consultant with approximately 12,000 employees and headquarters in various countries.

Table B1 presents the interviewees' description (PM1, PM2, PM3, and PM4) who work in companies of different characteristics. PM1 and PM4 work in the same company. PM1, PM2, and PM3 are graduates, and PM4 is postgraduate. Their experience ranges from 1 to 3 years and have participated in the management of 6 to 10 projects.

Table B1. Activities Supported by Information on the Developers' Work

ID	Company	Education	Experience	# Projects
PM1	A	Bachelor degree	1 year	6
PM2	C	Bachelor degree	1,5 year	8
PM3	B	Bachelor degree	3 years	10
PM4	A	MBA	1 year	8

In Figure 7, we presented the steps for collecting the PMs' opinions. The researchers defined questions and devised an electronic questionnaire. Then, PMs were asked to voice their opinions. We scheduled an individual interview with each PM and recorded their answers in the electronic questionnaire. Finally, we compiled the responses and included them in the discussion of the results.

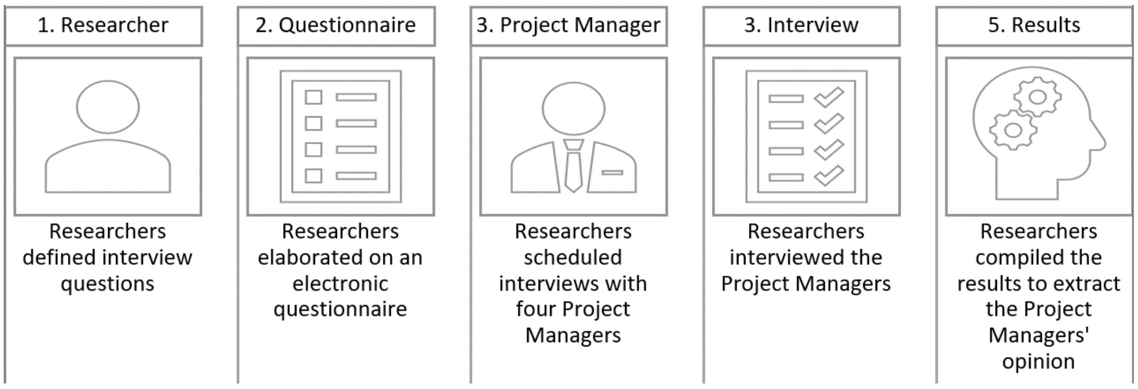
1. Researcher
Researchers defined interview questions
2. Questionnaire
Researchers elaborated on an electronic questionnaire
3. Project Manager
Researchers scheduled interviews with four Project Managers
3. Interview
Researchers interviewed the Project Managers
5. Results
Researchers compiled the results to extract the Project Managers' opinion

Figure 7. Steps to Collect PMs' opinions

[1] The questions, responses, and annotations are available at (in Portuguese): http://doi.org/10.5281/zenodo.3965805