Received July 9, 2020, accepted July 29, 2020, date of publication August 7, 2020, date of current version August 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3015017

Team Formation in Software Engineering: A Systematic Mapping Study

ALEXANDRE COSTA[1,2], FELIPE RAMOS[1,2], MIRKO PERKUSICH[1], EMANUEL DANTAS[1], EDNALDO DILORENZO[1], FERDINANDY CHAGAS[1], ANDRÉ MEIRELES[1], DANYLLO ALBUQUERQUE[1], LUIZ SILVA[1], HYGGO ALMEIDA[1], AND ANGELO PERKUSICH[1]

[1]Intelligent Software Engineering (ISE) Group, Federal University of Campina Grande (UFCG), Campina Grande 58429-900, Brazil
[2]Federal Institute of Paraíba (IFPB), Santa Luzia 58600-000, Brazil

Corresponding author: Alexandre Costa (alexandrecostapb@gmail.com)

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

ABSTRACT *Context:* Software team formation is an important project management activity. However, forming appropriate teams is a challenge for most of the companies. *Objective:* To analyze and synthesize the state of the art on the software team formation research. Additionally, we aim to organize the identified body of knowledge in software team formation as a taxonomy. *Method:* Using a Snowballing-based systematic mapping study, 51 primary studies, out of 2516, were identified and analyzed. We classified the studies considering the research methods used, their overall quality, and the characteristics of the formed teams and the proposed solutions. *Results:* The majority of the studies use search and optimization techniques in their approaches. Also, technical attributes are the most frequent type considered to build individuals' profiles during the team formation process. Furthermore, we proposed a taxonomy on software team formation. *Conclusion:* There is a predominant use of search-based approaches that combine search and optimization techniques with technical attributes. However, the adoption of non-technical attributes as complementary information is a tendency. Regarding the research gaps, we highlight the level of subjectivity in software team formation and the lack of scalability of the proposed solutions.

INDEX TERMS Team formation, software development, software engineering, software project planning, systematic mapping study.

I. INTRODUCTION

Software Project Management (SPM) includes a set of activities to deliver a software product and related artifacts (e.g., source code, models, test case, and documentation) to accomplish specific goals while satisfying a set of constraints [1]. A widely used model of SPM constraints, suggested in the Project Management Body of Knowledge (PMBOK), is known as the triple constraint that include cost, time, and scope. In the software industry, it is common to classify projects that satisfy the triple constraint as successful [2].

Delivering software projects adherent to the constraints is still a significant struggle for software companies [3]. In this context, human resource allocation plays a critical role [4] since people are directly involved in all software development activities. Therefore, forming an appropriate project team is an essential activity for SPM. For this study, we define an appropriate team as the most suitable configuration of the software development team (i) to properly perform activities of software development and (ii) to be compliant to the project's triple constraint.

Forming appropriate software teams is a challenge for most of the companies. Traditionally, the teams are formed based on the experience, subjective perception, and instinct of the managers, resulting in a non-automated, human dependent, and error-prone process, due to the problem's complexity [5]. The problem's complexity is associated with the number of considered criteria during the decision making process. Studies have presented relevant attributes to form a software development team. da Silva *et al.* [6] performed qualitative research and identified several attributes that managers consider to select team members such as technical abilities,

The associate editor coordinating the review of this manuscript and approving it for publication was Resul Das.

availability, project importance, individual cost, productivity, behavior, and personality [6]. Gilal *et al.* [7] investigated the use of members' personality types and genders to form software development teams. Tsai *et al.* [8] proposed a method for human resource selection based on the individual's productivity and salary. André *et al.* [4] presented a formal model for assigning human resources to software project teams using attributes such as programming language proficiency, graphic design abilities, product knowledge, teamwork, negotiation skills, proactivity, personality types, independence, and capacity to control. Chiang and Lin [9] proposed a framework for assisting a software company for making decisions regarding team formation based on project duration with labor skill and budget constraint.

Given the high diversity of attributes to be considered, the complexity of building appropriate teams rises fast with the increasing number of available candidates. Lappas *et al.* [10] showed that this problem is NP-hard. As it is widely known, solving NP-hard problem by evaluating all possibilities becomes fastly prohibitive. Hence, building appropriate teams without the support of proper tools can be a time-consuming, error-prone, repetitive, and exhaustive task.

Several solutions (e.g., methods, techniques, guidelines, and processes) have been proposed in the Software Engineering community and by industrial practitioners, aiming to address the software team formation problem. The solutions are based on technical and non-technical attributes (or both). For instance, Di Penta *et al.* [11] investigated the use of search-based techniques to support software project management, showing how they can be used to address problems of (i) allocating staff to teams, (ii) staffing level adjustment, (iii) reduction of project fragmentation, (iv) team formation based on the expertise of the candidates and (v) the required knowledge to deliver the given work package. Costa *et al.* [12] proposed a Genetic Algorithm combined with tag-based profiles to represent the developers' technical skills to form multiple software project teams. Finally, Majumder *et al.* [13] presented an approach to find socially close teams and a division of the task among team members, aiming to cope with allocation overload problems.

According to Petersen *et al.* [14], as a research area matures, the number of reports and results made available also increases, and it becomes important to summarize and provide an overview. Therefore, given the relevance of the software team formation problem and the diversity of available solutions, it raises the need to structure the field systematically. To the best of our knowledge, we did not find any exploratory studies directly related to software team formation in the literature.

Aiming to address this research gap, we performed a Systematic Mapping Study (SMS) [14] with the purpose to characterize the state of the art on software team formation. For doing so, we conducted the SMS following the guidelines given by Wohlin [15] for the Snowballing procedure. As a result, we analyzed 2510 papers, resulting in 51 relevant, which we further classified into 40 studies. Moreover, we used the data collected in this study to propose a taxonomy to organize the knowledge on the research topic.

The results discussed in this SMS can benefit:

- researchers who are interested in comprehending the state of the art on the software team formation. The systematic classification of the existing research provides a body of knowledge for deriving hypotheses and identifying new areas for future research;
- practitioners who may be interested in understanding the reported solutions in terms of methods and techniques to support the software team formation.

The remainder of this article is structured as follows. Section II presents the methodology used to conduct this SMS. Section III discusses the results of the SMS. Section IV details the proposed taxonomy. Section V shows the threats to validity. Section VI presents the final remarks as well as the main future works opportunities.

II. METHODOLOGY

We executed a Snowballing-based SMS [14], following the guidelines presented by Wohlin [15]. In what follows, we detail the study's protocol.

A. RESEARCH QUESTIONS

The objective of this SMS is to provide an overview of the state of the art on software team formation research. Therefore, we formulated the Research Questions (RQs) shown in Table 1.

B. SEARCH METHOD

The Snowballing procedure focuses on the analysis of a seed set of papers, regarding the papers' reference (backward snowballing) and citation (forward snowballing) lists. Therefore, an essential task to apply the Snowballing is to determine the seed set. For this purpose, we established a search string, discussed in Section II-B1, and used Google Scholar as the reference search engine. Wohlin [15] recommends Google Scholar to avoid bias in favor of any specific publisher. Afterward, we used the selection criteria shown in Section II-C to analyze the papers identified through the search. As a result, we defined the seed set and started the snowballing iterations. Figure 1 shows an overview of the executed procedure.

1) SEARCH TERMS

To define the search terms, we selected a set of previously known papers related to the research topic. Then, we extracted the relevant terms from these papers' titles, abstracts, and keywords. A very common term in the context of software development is *team formation*, however, other terms are also frequently used in the related literature such as *team allocation*, *team selection*, *team composition*, and *team configuration*. Additionally, the term *human resource allocation* is also related to papers on the scope of this study. Thus, we combined it with the previous ones to complement the search. In what follows, we present the search string:

TABLE 1. Research questions of the study.

Id	Research Question	Objective
RQ-1	What is the state of the art on software team formation research?	To provide an overview of team formation research in the context of software development.
RQ-1.1	What research methods have been used in software team formation studies?	To categorize available team formation studies in the context of software development, according to their research questions, results, and validation strategies.
RQ-1.2	What are the characteristics of the recommended software teams?	To identify the characteristics of the teams based on the target software development methodology, the geographical distribution, the role composition, and the types of attributes considered to form the teams.
RQ-1.3	What are the characteristics of the proposed solutions?	To identify the characteristics of the proposed solutions based on their objectives, category of the techniques or algorithms, outputs, and how their results are assessed.
RQ-1.4	What is the quality of the software team formation research?	To assess the quality of team formation research in the context of software development by examining the scientific rigor.
RQ-2	What are the trends and gaps in software team formation research?	To identify trends and gaps in team formation research in the context of software development.

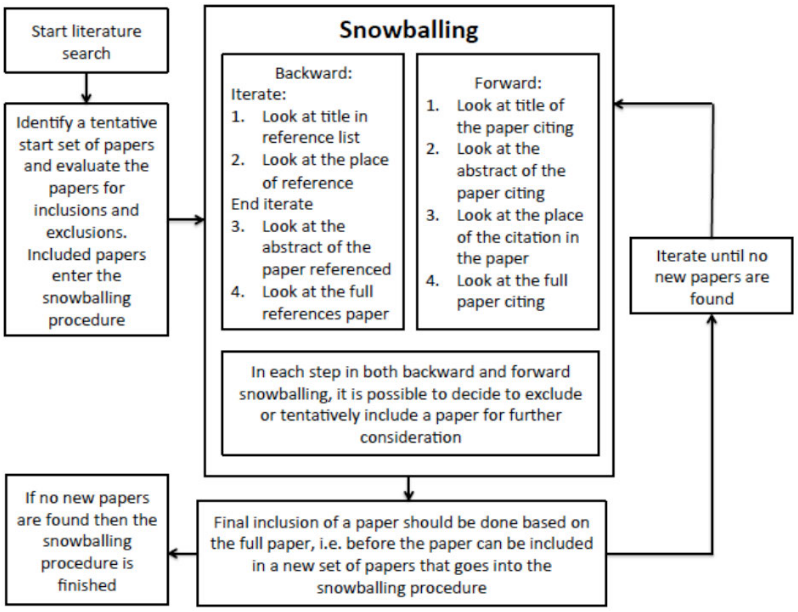

Start literature search
Identify a tentative start set of papers and evaluate the papers for inclusions and exclusions. Included papers enter the snowballing procedure
Snowballing
Backward:
Iterate:
1. Look at title in reference list
2. Look at the place of reference
End iterate
3. Look at the abstract of the paper referenced
4. Look at the full references paper
Forward:
1. Look at title of the paper citing
2. Look at the abstract of the paper citing
3. Look at the place of the citation in the paper
4. Look at the full paper citing
In each step in both backward and forward snowballing, it is possible to decide to exclude or tentatively include a paper for further consideration
Iterate until no new papers are found
If no new papers are found then the snowballing procedure is finished
Final inclusion of a paper should be done based on the full paper, i.e. before the paper can be included in a new set of papers that goes into the snowballing procedure

FIGURE 1. Snowballing procedure presented by Wohlin [15].

("team formation" OR "team allocation" OR "team selection" OR "team configuration" OR "team composition" OR "human resource formation" OR "human allocation resource" OR "human resource selection" OR "human resource configuration")

Following the guidelines presented in [15], we set the search parameters to return papers based on the title match, with at least one of the terms, and only for 2016, the year before the beginning of this SMS. The year constraint does not negatively impact the study since previous papers are supposed to be found during the backward process. In the same way, papers from later years should appear during the forward process.

C. SELECTION CRITERIA

The selection criteria (i.e., inclusion and exclusion criteria) considered in this study were divided into three phases: general exclusion, basic criteria, and advanced criteria.

1) GENERAL EXCLUSION

In this phase, we aimed to eliminate irrelevant studies for our SMS. We considered irrelevant the studies that met at least one of the criteria listed following:

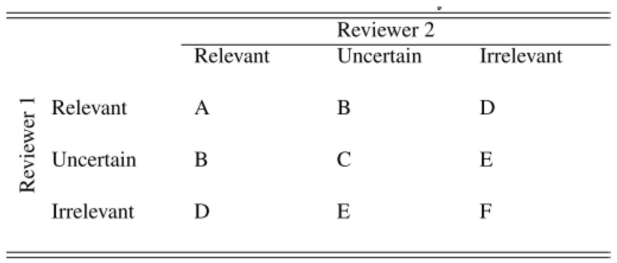
1) Papers published before the 21st century (i.e., before 2001).
2) Papers not written in English.
3) Duplicate or already analyzed papers.
4) Secondary studies.
5) Short papers or expanded abstracts.
6) Technical reports, dissertations, thesis, or books.
7) Papers not published in journals, magazines, conferences, or workshops.
8) Primary studies that do not present a method, procedure, model, methodology, (semi-)automated support, guidelines, or any criteria that explicitly provide means to form software development teams.

2) BASIC CRITERIA

In this phase, we randomly chose peer reviewers to analyze the remaining papers' titles, abstracts, and keywords from the general exclusion phase. Therefore, we classified them according to the procedure defined in [16] as:

- **Relevant:** papers related to software team formation.
- **Irrelevant:** papers not related to software team formation.
- **Uncertain:** papers in which the information available in the titles, abstracts, and keywords are insufficient or inconclusive to classify the papers as relevant or irrelevant.

Ali *et al.* [16] suggest six levels of agreement between reviewers, presented in Figure 2. The *A* and *B* levels mean that at least one reviewer classified the paper as relevant, and consequently, it will pass to the next phase. *B* indicates that a reviewer classified as uncertain, but the paper will be included so we can minimize the risk of discarding any relevant study. This decision has no negative impact on the final set of papers because if it is not relevant, it will be discarded in the next phase. The *C* level means none of the reviewers are sure about their classifications, and an additional assessment is required. In this case, both reviewers do a careful reading, in which they start from the introduction of the paper and, if not sufficient, they go to the conclusion section. *D* and *E* indicate that reviewers disagreed, and they must discuss the reasons that led them to that classification. If there is no consensus, a third reviewer is called in to make a final decision. Finally, *F* level means the reviewers agree that the paper is not relevant, and it must not be included.



		Reviewer 2		
		Relevant	Uncertain	Irrelevant
Reviewer 1	Relevant	A	B	D
	Uncertain	B	C	E
	Irrelevant	D	E	F

FIGURE 2. Levels of agreement between reviewers. Image extracted from [16].

3) ADVANCED CRITERIA

In this phase, we fully evaluated the papers by performing as complete as possible reading. For this purpose, once more, we randomly chose two reviewers (data extractor and data checker) to evaluate the remaining papers from the previous phase. The data extractor was responsible for evaluating the quality of the papers (Section II-D) and for extracting data from them (Section II-E).Then, the data checker reviewed all the extracted data to confirm their correctness, following the guidelines suggested by Brereton *et al.* [17] and Staples and Niazi [18].

D. QUALITY ASSESSMENT

To determine the quality of the papers, we considered 11 criteria proposed in [19]. After a complete reading, the data extractor evaluated each criterion using a Boolean scale (i.e., 0 or 1). Next, the data checker reviewed the answers and, in the case of divergence, the reviewers discussed with each other and tried to reach a consensus; otherwise, a third reviewer would be requested. The 11 criteria can be observed as follow:

1) Does the paper represent scientific research?
2) Are the research objectives clearly defined?
3) Is there an adequate description of the context in which the research was carried out?
4) The research design was appropriate to address the aims of the research?
5) Was the recruitment strategy adequate to the research objectives?
6) Was there a control group to compare the treatments?
7) Was the data properly collected to address the research problem?
8) Was the data analysis sufficiently rigorous?
9) Was the relationship between the researcher and participants considered to an adequate degree?
10) Is there a clear definition of the findings with credible results and justified conclusions?
11) Does the study provide value for researchers or practitioners?

E. DATA EXTRACTION

To acquire quantitative and qualitative data to answer our research questions accurately, we extracted the following data:

1) Document title.
2) Authors.
3) Publication year.
4) Publication channel.
5) Research question type (Table 2).
6) Research result type (Table 3).
7) Research validation type (Table 4).
8) For which software development methodology the study aims to form teams.
9) The geographical distribution of the teams.
10) The role composition of the teams.
11) The type of attribute considered to form teams
12) The type of attribute assignment.
13) The technique used to form teams.
14) The objective of the proposed solution.
15) The output of the proposed solution.
16) The type of assessment used to validate the proposed solution.

The items *research question type*, *research result type* and *research validation type* were provided from the classification presented by Shaw [20] and can be seen in the Tables 2, 3 and 4, respectively.

III. RESULTS

This section outlines the results of the SMS regarding the research questions. Section III-A presents the results of the search process and of classifying the selected studies given

TABLE 2. Research questions types. Classification extracted from [20].

Research question	Description
Method or means of development	How can we do/create/modify/evolve (or automate doing) X? What is a better way to do/create/modify/evolve X?
Method for analysis or evaluation	How can I evaluate the quality/correctness of X? How do I choose between X and Y?
Design, evaluation, or analysis of a particular instance	How good is Y? What is property X of artifact/method Y? What is a (better) design, implementation, maintenance, or adaptation for application X? How does X is compared to Y? What is the current state of X / practice of Y?
Generalization or characterization	Given X, what will Y (necessarily) be? What, exactly, do we mean by X? What are its important characteristics? What is a good formal/empirical model for X? What are the varieties of X, how are they related?
Feasibility study or exploration	Does X even exist, and if so what is it like? Is it possible to accomplish X at all?

TABLE 3. Research result types. Classification extracted from [20].

Research result	Description
Procedure or technique	New or better way to do some task, such as design, implementation, maintenance, measurement, evaluation, selection from alternatives; includes techniques for implementation, representation, management, and analysis; a technique should be operational as a procedure. Advice, checklists and guidelines will be excluded.
Qualitative or descriptive model	Structure or taxonomy for a problem area; architectural style, framework, or design pattern; non-formal domain analysis, well-grounded checklists, well-argued informal generalizations, guidance for integrating other results, well-organized interesting observations.
Empirical model	Empirical model based on observed data.
Analytic model	Structural model that permits formal analysis or automatic manipulation.
Tool or notation	Implemented tool that embodies a technique; formal language to support a technique or model (should have a calculus, semantics, or other basis for computing or doing inference)
Specific solution, prototype, answer, or judgment	Solution to application problem that shows application of SE principles – may be design, prototype, or full implementation; careful analysis of a system or its development, result of a specific analysis, evaluation, or comparison.
Report	Interesting observations, rules of thumb, but not sufficiently general or systematic to rise to the level of a descriptive model.

the publication channel. The subsequent sections present proper answers to the study's research questions. We released a link with supplementary material of the research.[1]

A. GENERAL RESULTS

During the Snowballing procedure, a total of 2516 papers were evaluated with the selection criteria defined in Section II-C. We analyzed 112 papers to build the start set and chose seven relevant ones. Next, we performed five Snowballing iterations and evaluated 2404 additional papers.

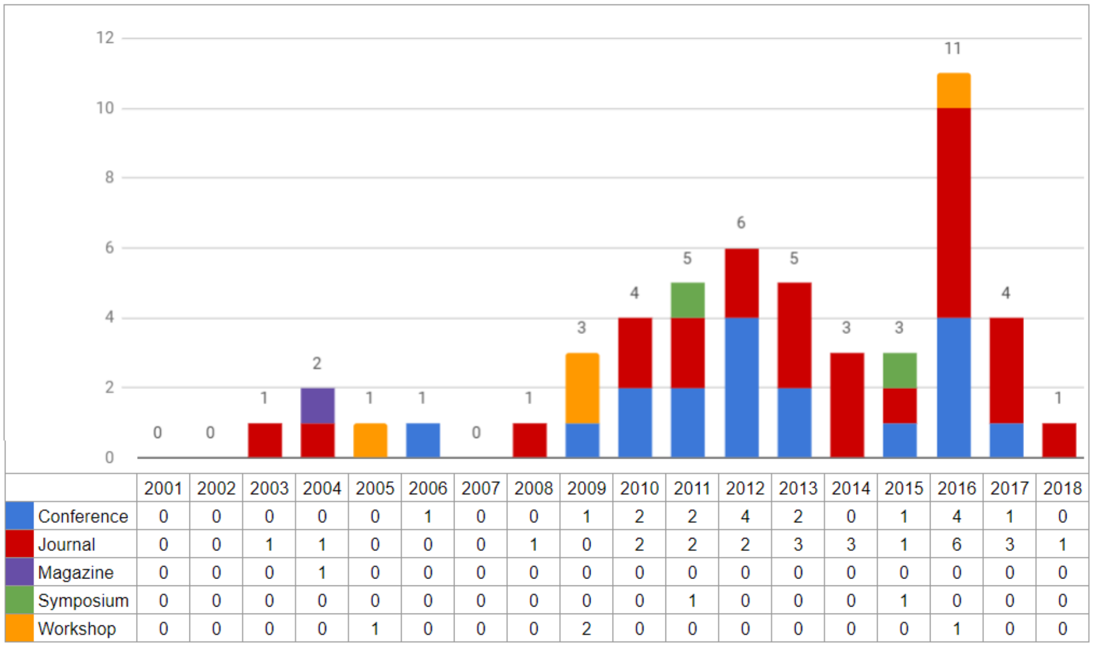
[1]https://bit.ly/2lB44LA

TABLE 4. Research validation types. Classification extracted from [20].

Research validation	Description
Analysis	I have analyzed my result and find it satisfactory through rigorous analysis. - For a formal model ... rigorous derivation and proof - For an empirical model ... data on use in controlled situation - For a controlled experiment ... carefully designed experiment with statistically significant results.
Evaluation	Given the stated criteria, my result... - For a descriptive model ... adequately describes phenomena of interest ... - For a qualitative model ... accounts for the phenomena of interest ... - For an empirical ... model is able to predict ... because ..., or ... generates results that fit actual data ... Includes feasibility studies, pilot projects
Experience	My result has been used on real examples by someone other than me, and the evidence of its correctness/usefulness/effectiveness is ... - For a qualitative model ... narrative - For an empirical model or tool ... data, usually statistical, on practice - For a notation or technique ... comparison of systems in actual use
Example	Here's an example of how it works on - For a technique or procedure ... a "slice of life" example based on a real system. - For a technique or procedure ... a system that I have been developing. - For a technique or procedure ... a toy example, perhaps motivated by reality. The "slice of life" example is most likely to be convincing, especially if accompanied by an explanation of why the simplified example retains the essence of the problem being solved. Toy or textbook examples often fail to provide persuasive validation, (except for standard examples used as model problems by the field).
Persuasion	I thought hard about this, and I believe passionately that ... - For a technique ... if you do it the following way, then ... - For a system ... a system constructed like this would ... - For a model ... this example shows how my idea works. Validation purely by persuasion is rarely sufficient for a research paper. Note, though, that if the original question was about feasibility, a working system, even without analysis, can suffice.
Blatant assertion	No serious attempt to evaluate result. This is highly unlikely to be acceptable.

As a result, we selected an additional 44 relevant papers. At the end of the search process, we identified 51 relevant papers, published between 2001 and 2018. Table 5 shows the distribution of the papers' evaluation throughout the iterations. In each iteration, we performed the backward and forward Snowballing procedures. Table 8 (Appendix) presents basic information of each included paper, which consists of paper/study identification, paper title, Snowballing phase in which the paper was found, and the paper reference.

Figure 3 shows the distribution of papers according to the publication channel from 2001 to 2018. Out of 51 papers, 51.00% are published in journals, 35.30% in conferences, 7.80% in workshops, 3.90% in Symposium, and 2.00% in magazines. Observing Figure 3, we notice an increasing number of publications over the years but in a non-linear way. In 2001, 2002, and 2007, we did not find any published papers. Before 2008, the number of papers is smaller



	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
🟦 Conference	0	0	0	0	0	1	0	0	1	2	2	4	2	0	1	4	1	0
🟥 Journal	0	0	1	1	0	0	0	1	0	2	2	2	3	3	1	6	3	1
🟪 Magazine	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
🟩 Symposium	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
🟧 Workshop	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	1	0	0

FIGURE 3. **Distribution of the papers according to the publication channel from 2001 to 2018.**

TABLE 5. **Statistics of the Snowballing iterations.**

Iteration	References	Citations	After general exclusion	After basic criteria	After advanced criteria
1	189	4	98	17	11
2	370	476	580	30	18
3	473	295	135	48	13
4	444	61	37	2	2
5	50	42	12	0	0
Total	**1526**	**878**	**862**	**97**	**44**

compared to the last decade (2008-2018). We believe there is a recent growing interest in the research area, with an evident peak in 2016. The reduced number of papers identified in 2018 is due to the search process, in which the last Snowballing iteration was executed in February 2018. Therefore, the reader should not interpret the reduced number of identified papers in 2018 as a lack of published papers.

Since multiple papers might be published as results of a research study, to avoid bias in answering our research questions, we classified the included papers into studies. The classification was based on analyzing the papers' authors. The paper chose to represent the study on our analysis was based on the most recently published research (by year) and, in the case of a tie, the one which obtained the highest score in the quality assessment. For instance, the papers *P*04 and *P*05 were considered the same study (*S*04). The authors first published the research in a conference and then submitted an extended version, which was published in a journal. Although both papers were published in the same year, we chose *P*04 to represent the study *S*04, because it achieved a higher quality score.

B. RQ-1.1: WHAT RESEARCH METHODS HAVE BEEN USED IN SOFTWARE TEAM FORMATION STUDIES?

According to Figure 4, most studies seek to answer questions about *Method or means of development* (85.00%), followed by *Method for analysis or evaluation* (12.50%) and *Design,*

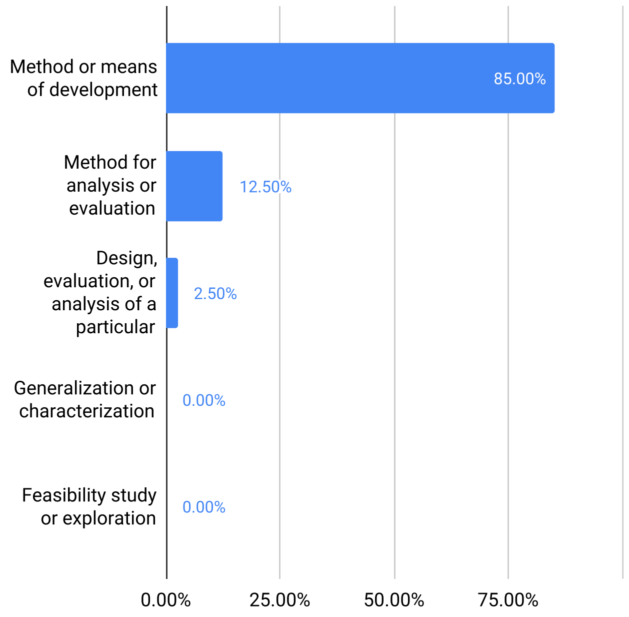


FIGURE 4. **Classification of the studies according to the research question type.**

evaluation, or analysis of a particular instance (2.50%). We believe *Method or means of development* outstands the other categories due to the nature of the identified studies: solve a software engineering problem, namely, the team formation problem in the context of software development.

Figure 5 shows the classification of the studies regarding the research result types. 65.00% presented a *Procedure or technique*, 17.50% a *Tool or notation*, 10.00% an *Empirical model*, 2.50% a *Qualitative or descriptive model*, 2.50% a *Report*, and 2.50% a *Specific solution, prototype, answer, or judgment*. The higher percentage of *Procedure or technique* is justified because most studies present solutions to automatize the team formation process, which endorse the findings of the previous research question (R-1.1).

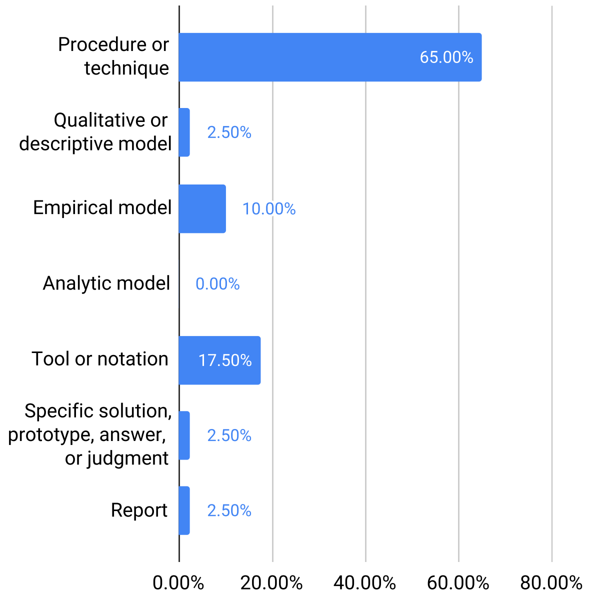


FIGURE 5. Classification of the studies according to the research result type.

Figure 6 presents the distribution of the studies according to the research validation types. Most of them used *Analysis* (47.50%), followed by *Evaluation* (32.50%), *Example* (15.00%) and *Blatant assertion* (2.50%). In studies that use *Analysis* as the validation type, the authors usually choose specific metrics to check the quality of the achieved results.

On the other hand, studies that use *Evaluation* count on the presence of specialists, usually managers who are familiar with the software projects. These managers manually choose their teams and then compare them to those provided by the proposed solution or, in other cases, they qualitatively evaluate the teams provided by the solution. Table 9 (Appendix) presents the classification of each included papers, regarding research data.

C. RQ-1.2: WHAT ARE THE CHARACTERISTICS OF THE RECOMMENDED SOFTWARE TEAMS?

To answer this question, we focused on four points we believe represent the main characteristics of the recommended teams: (i) to identify for which software development methodology

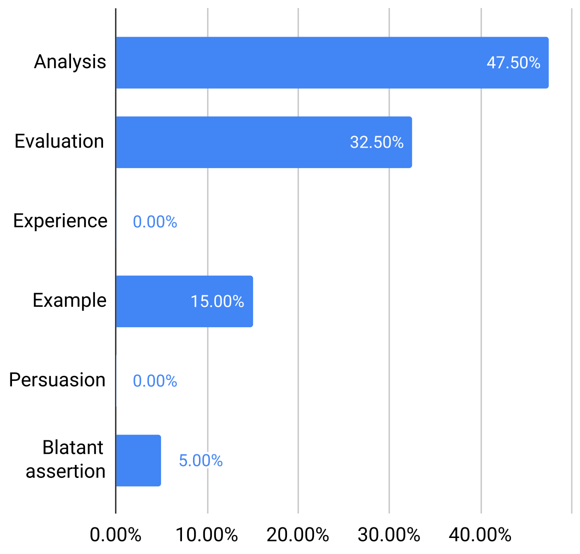


FIGURE 6. Classification of the studies according to the research validation type.

the researches aims to form teams; (ii) to determine how the formed teams are geographically distributed; (iii) identify the role composition of the formed teams; (iv) identify the types of attributes used to form teams and how their values are assigned.

Software development methodologies are usually classified into plan-driven or agile [21]. In the former, all activities are planned in detail, and progress is evaluated in comparison with the initial planning. In the latter, planning is gradual, and it is easier to change the process to reflect customers' changing needs. Depending on the target development methodology, the desired characteristics of the teams may vary. For instance, plan-driven teams usually are document-oriented and composed of specialists such as software architects and system analysts.

Conversely, agile teams prioritize communication over documentation; thus, they can be more collaborative. 15.00% of the identified studies claim to form agile teams. However, the majority (85.00%) does not specify the target methodology. Scrum and Extreme Programming are the only agile methods specified, but just in two studies. Most of the proposed solutions to form agile teams use data from agile projects as input. However, they do not specify which elements their solutions incorporate to guarantee that the teams will be truly agile.

Global software development has become an important model for developing and delivering software products. Software companies continue to expand and disperse geographically in search of high-skilled human resources, lower project cycle times, and to reduce costs [22]. Conversely, physical separation may difficult team members' ability to properly communicate and coordinate their work and stay aware of other members' activities [23]. 10% of the identified studies proposed solutions to form global teams, i.e., teams in which members are geographically distributed. These approaches typically use data from open-source development platforms such as GitHub or Sourceforge to validate the results. These platforms make possible professionals, from different places around the world, work together as a team in several projects.

Software teams can be composed of multiple roles, such as developers, architects, analysts, testers, designers, and others. Figure 7 presents the roles identified during this research. In 15.00% of the studies, the teams are composed of *Developers* only, which is common in agile development. For instance, in Scrum, the teams are cross-functional, i.e., all members must have all the competencies needed to achieve their goals with external dependence [24]. 15.00% of the studies explicitly divide the team into *Team leader and team members*. Some researchers [25]–[27] argue that the leader role is directly related to the success or failure of the team. Appropriate teams demand good leadership to manage and facilitate team interaction and for assisting the members during conflicts. 22.50% of the studies recommend teams composed of *Multiple roles* such as quality manager, requirement engineer, support specialist, configuration manager, system analyst, and others. Generally, plan-driven teams have specialists to focus their work on specific phases of the development cycle. The majority of the studies (47.50%) does not specify any role in their solutions.

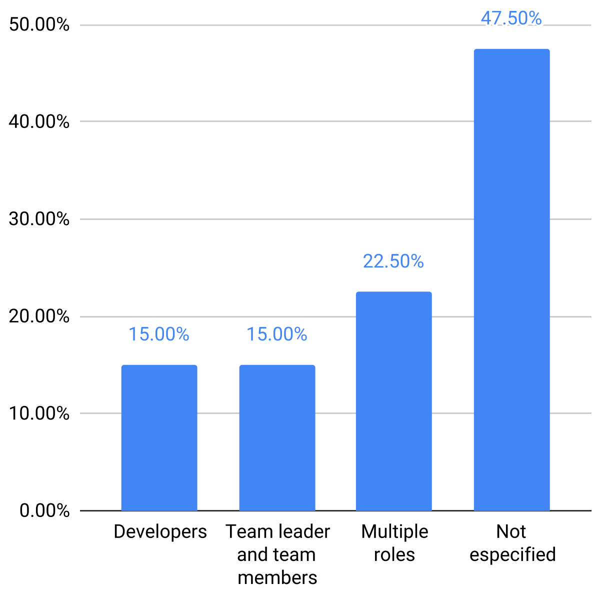


FIGURE 7. Team role compositions.

An important step of the team formation process is defining the attributes that will be used to find the most appropriate team. Normally, more than one type of attribute is used in the process. Figure 8 shows five types considered in the primary studies. Most of the studies consider *Technical* attributes such as experience in programming, network techniques, software design, quality control, database techniques, and others. The second most frequent type of attribute is the *Individual cost* (35.00%). The project over budget is one of the main reasons for project failure [28] and remains a great concern in

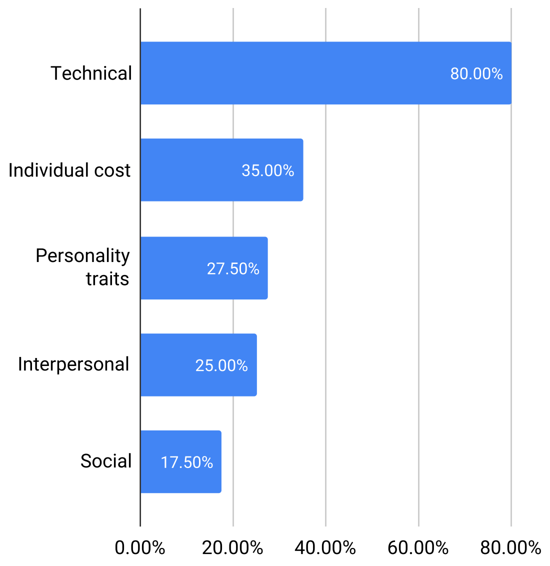


FIGURE 8. Types of attributes used during the team formation process.

software companies. Therefore, a significant amount of solutions seek to achieve teams that minimize the financial cost. Usually, *Individual cost* is calculated using the salary per hour of each candidate member. *Personality traits* are present in 27.50% of the studies. This type of attribute is originated from *Psychometric instruments* such as Belbin's Team Roles [29] and Myers-Briggs Type Indicator [30]. 25.00% of the studies use *Interpersonal* attributes such as communication, positive criticism, creativity, emotional intelligence, leadership, and others. Generally, in these studies, *Interpersonal* attributes are used together with other types. 17.50% of the studies use *Social* attributes that represent metrics that can establish links between members. For instance, the number of projects and the amount of time they worked together, their level of friendship, the number of coworker friends they have in common, and others. It is important to notice that the sum of percentages exceeds 100% in Figure 8 because one study can consider more than one type of attribute.

Another important step of the team formation process is to define how the attribute values are assigned, which can increase or reduce the process subjectivity. Figure 9 presents the types of assignments identified in this SMS. 30.00% of the studies use *Historical data* to define the attributes' values. These studies retrieve data from private or public software development repositories. In the first case, the data is usually retrieved from the company's repository, which stores data of projects that were (or still are) executed in the company. In the second case, the data is retrieved from online repositories such as GitHub or Sourceforge. *Expert judgment* is the second most frequent at 22.50%. In this type of assignment, a set of experts is responsible for quantifying the attributes. Generally, project managers are chosen because they are familiar with the candidate members. 17.50% of the studies use *Psychometric instruments*, which are questionnaire-based tools to measure candidate members' personality traits. *Self-assessment* is used in 17.50% of the

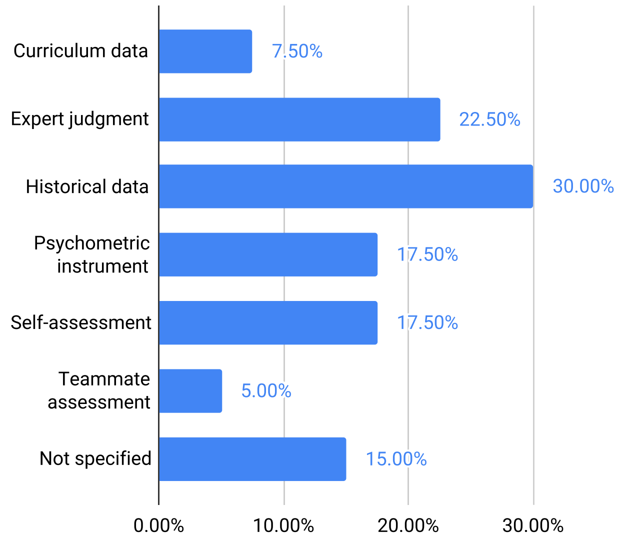


FIGURE 9. Types of assignments of the attributes used during the team formation process.

studies and consists of candidate members assigning values for their attributes. For instance, a questionnaire containing lists of predefined attributes is created, and the candidate members are invited to answer based on how they see themselves. 5.00% of the studies use *Teammate assessment*, which is similar to *Self-assessment*, but instead of evaluating themselves, the candidate members are asked to assess their colleagues. *Expert judgment*, *Self-assessment*, and *Teammate assessment* are commonly used in 360° feedback, which is a model for employee assessment and performance improvement. 7.50% of the studies use the *Curriculum data* to assign the attributes' values. In this case, the data is extracted and formatted to be used as input to the proposed solution. 15.00% of the studies do not specify how the values were assigned. Analogous to the types of attributes, studies can use more than one type of assignment.

D. RQ-1.3: WHAT ARE THE CHARACTERISTICS OF THE PROPOSED SOLUTIONS?

To answer this question we defined four points to represent aspects related to the proposed solutions: (i) identify the objectives to be achieved using the proposed solution; (ii) identify the category of the technique used in the proposed solution; (iii) identify the outputs of the proposed solution; (iv) identify how the proposed solution's results are assessed.

According to Figure 10, most of the proposed solutions (67.50%) aim to *Maximize project's requirements*. This objective consists of finding a team in which its member's skills match the target project requirements. Traditionally, during the team formation process, dozens or even hundreds of candidate members with several different skills are evaluated. Thus, without automated support, it is difficult to find the most appropriate team. 27.50% of the studies aims to *Improve work relationships*, which means they try to find a team where its members could easily interact with each other to create a harmonic work environment. This objective

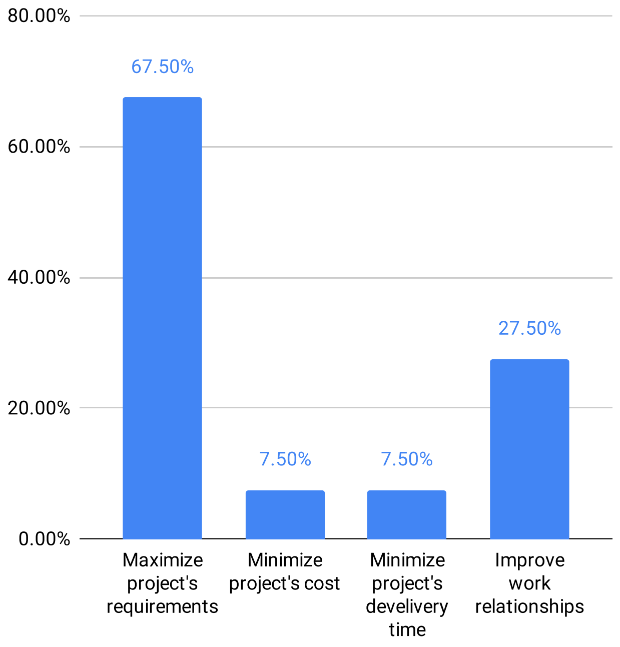


FIGURE 10. Objectives of the studies.

is particularly frequent in psychometric instrument-based solutions. In 7.5% of the studies, the main purpose is to find a team that can *Minimize project's cost* by selecting members based on their individual cost. Similarly, 7.50% of the solutions seek to form a team that can *Minimize project's delivery time*, i.e., they focus on delivering the final product as fast as possible to release the team members to work in other projects.

We identify almost 30 different techniques used in the proposed solutions, which were classified into five categories (Figure 11). The most popular is *Search and optimization* (62.50%), which includes Genetic Algorithms (GAs), Dynamic Programming, and Backtracking (i.e., Branch and Bound) algorithms. In particular, GAs represent the most commonly used technique, present in 20.00% of all studies.

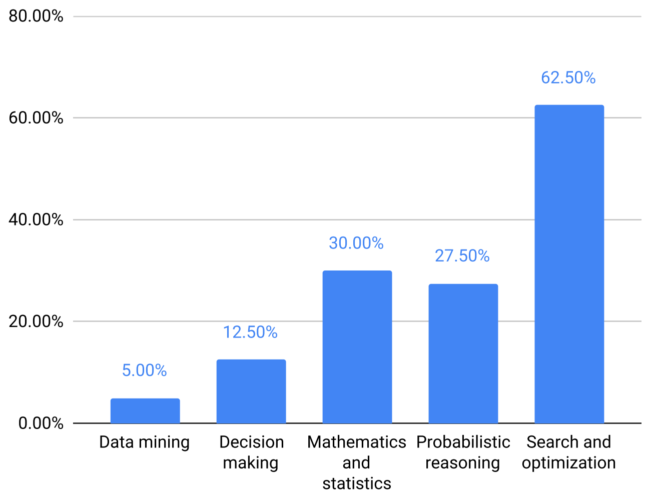


FIGURE 11. Categories of the techniques used in the proposed solutions.

Mathematics and statics appear in second place (30.00%), and it is composed of regression models, distance measures, statistical methods, and others. *Probabilistic reasoning* is the third most frequent category, present in 27.50% of the studies. Within this category, Fuzzy Logic and Bayesian networks are the frequently used techniques, and they are applied to handle the uncertainty related to the generation of individuals' profiles by managers. Within the fourth most frequent category, *Decision making* (12.50%), we identified Gray decision theory, Analytic Hierarchy Process, and Ontology techniques. In general, studies using these techniques do not attempt to automate tasks associated with forming development teams but rather guide the decision-maker in selecting team members. Tied with decision making, we identified *Data mining*, which consists of techniques based on clustering and association rules.

The most frequent output of the proposed solutions (45.00%) is *Multiple teams* (Figure 12). Although this type of output can vary from study to study. For instance, in some studies, the output can be multiple teams, one for each target project, and no member can be part of two different teams. In other studies, the output can be multiple teams for the same project, but with different members in each team. In this case, the project manager can choose the best one. Additionally, some studies output multiple teams for multiple projects, and members can be part of more than one team. 22.50% of the solutions outputs a *Ranked list* from which project managers can choose candidate members to composed their teams. This type of output allows the managers to choose the top-k candidates or even those that are not necessarily on the top of the list. 15.00% of the solutions output a *Single team* and the project manager just decide to accept it or not. In 10.00% of the solutions, the output is a *Set of rules* which are automatically generated from data mining techniques such as the Apriori algorithm. 5.00% of the studies output a *Set of criteria* that consists of guidelines based on expert experience or empirical research results. 2.50% of the studies do not specify what the output of their solutions is.

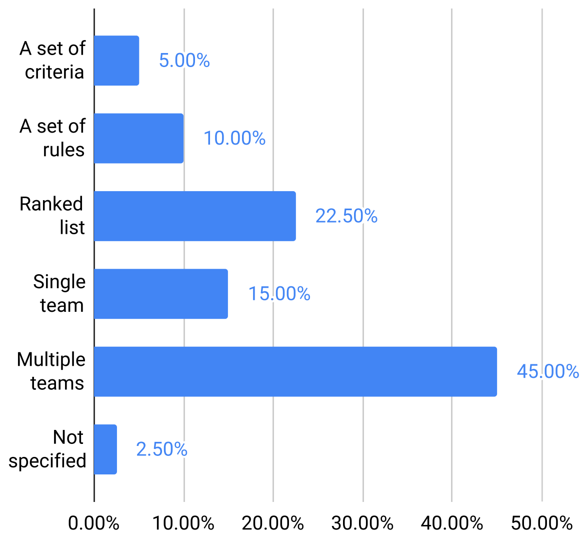


FIGURE 12. Outputs of the proposed solutions.

Figure 13 shows the types of assessments used to validate the proposed solution's results. 42.50% of the studies use *Predefined metrics*, which consists of verifying if the results match expected thresholds such as cost, project duration, and others. In 40.00% of the studies, the proposed solution's results were assessed by *Expert judgment*. The experts usually are project managers and the assessment can occur in two ways: (i) the experts use their experiences and knowledge to analyze the solution's results and then they provide feedback; or (ii) the experts define an optimal team formation which is compared to the recommendation of the proposed solution using accuracy metrics such as Precision and Recall. 10.00% of the studies used *Synthetic data* to create examples to demonstrate how useful the solution results are. 7.50% of the studies did not assess the proposed solution's results.

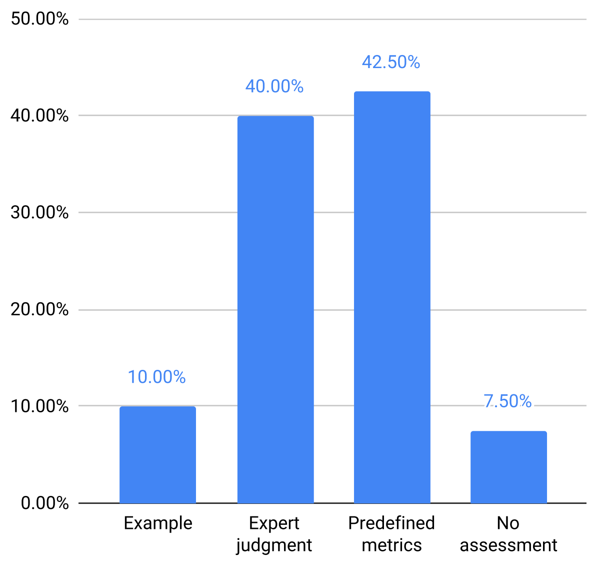


FIGURE 13. Types of assessments used to validate the proposed solution results.

E. RQ-1.4: WHAT IS THE QUALITY OF THE SOFTWARE TEAM FORMATION RESEARCH?

Figure 14 presents the overall score concerning the quality assessment described in Section II-D. Two criteria have low frequency: *Control Group* and *Reflexivity*. Studies that used *Control Group*, conducted experiments in which teams were formed using different methods such as the company's and the authors' proposed solution. In these cases, the main objective was to verify which teams were more appropriate over time. This practice is not frequently used by the studies, because companies usually have low interest in adopting innovative and not validated solutions in such a high-risk activity such as forming a team. Regarding *Reflexivity*, we notice that most studies do not critically analyze their results. For instance, they do not identify threats to the validity of the research.

Concerning the remaining criteria, 75.00% of the studies scored, which can be considered a positive indicator.

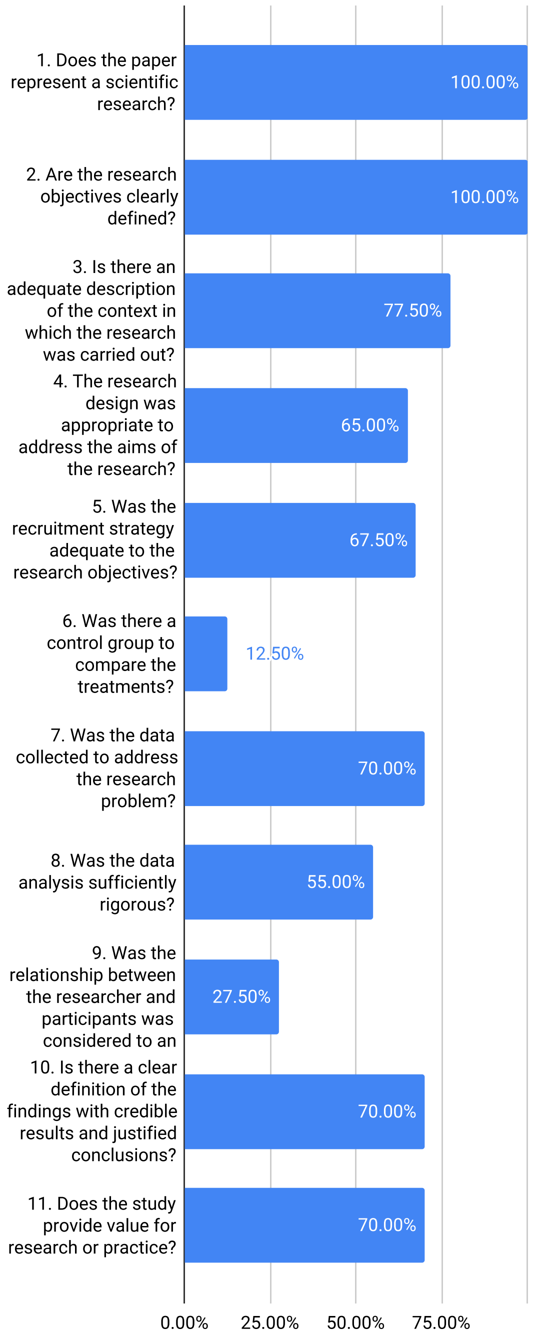


FIGURE 14. **Quality score achieved by the studies per criteria.**

Table 11 (Appendix) shows the full quality score of each included paper.

F. RQ-2: WHAT ARE THE TRENDS AND GAPS IN SOFTWARE TEAM FORMATION RESEARCH?

As a trend, we highlight the growing number of search-based approaches to mitigate the team formation problem in the context of software development. Figure 15 presents the distribution of the studies' technique categories from 2001 to 2018. We notice a high number of solutions based on *Search and optimization* techniques in the last decade. This tendency is justified because the team formation problem is trivially treated as a search and optimization one, and there are known approximation algorithms (e.g., Greedy and Dynamic Programming) and meta-heuristic algorithms (e.g., GAs and Simulated Annealing) that are useful to handle the NP-hard problems, which is the case for the team formation problem. If used properly, they provide solutions with acceptable running time complexity and good enough results to assist in the decision-making process [31].

Figure 16 presents the distribution regarding the types of attributes over the past years. The use of technical attributes has been overcoming the other ones through the last decade, which matches the adoption of *Search and Optimization* in a similar period. This behavior can be confirmed by the high number of studies using technical attributes and *Search and Optimization* techniques (see Figure 17). We believe the reason for this phenomenon is that this type of attribute is easier to identify, measure, and collect since technical characteristics are more common in technological-based fields such as Software Engineering.

On the other hand, there is also an increasing number of studies using non-technical attributes, also known as soft skills (e.g., personality traits, social, and interpersonal attributes). Some researchers argue that only the technical ones are not sufficient to form appropriate teams. For instance, Ahmed *et al.* [32] conducted a survey in the industry about the importance of *soft skills* in various roles of software development activities. The results indicate that this type of attribute is critical in the software industry, having communication skills as the top required. The main drawback of adding *soft skills* to the team formation process is the cost to build the individuals' profiles, since soft skills are more challenging to identify, measure, and quantify. Moreover, we did not find any research accurately comparing the costs and benefits of adding soft skills.

Feldt *et al.* [33] presented ten levels of automation (Table 6) for Artificial Intelligence (AI) techniques applied to SE. This classification conveys how different human operators and systems/solutions should cooperate by sharing the control of determining and selecting options to perform tasks. Thus, we decided to use this classification to provide an overview of the level of automation of the proposed solutions. Although not all studies use AI techniques, we performed the classification considering the computational technique used during the team formation process. Figure 18 shows the distribution of the studies according to the level of automation. Level one (10.00%) correspond to studies that did not use any computational technique. Instead, the authors proposed guidelines or a set of criteria based on surveys or expert experiences. Studies in level two (2.50%) propose a solution that generates rules to form teams, based on analyzed data. Studies in level three (20.00%), do not provide a team, but a ranked developer list based on specific criteria, that assist managers in selecting the members to their teams. Level four (67.50%) is composed of studies that present a computational technique that suggests an optimal team configuration, and the manager must accept

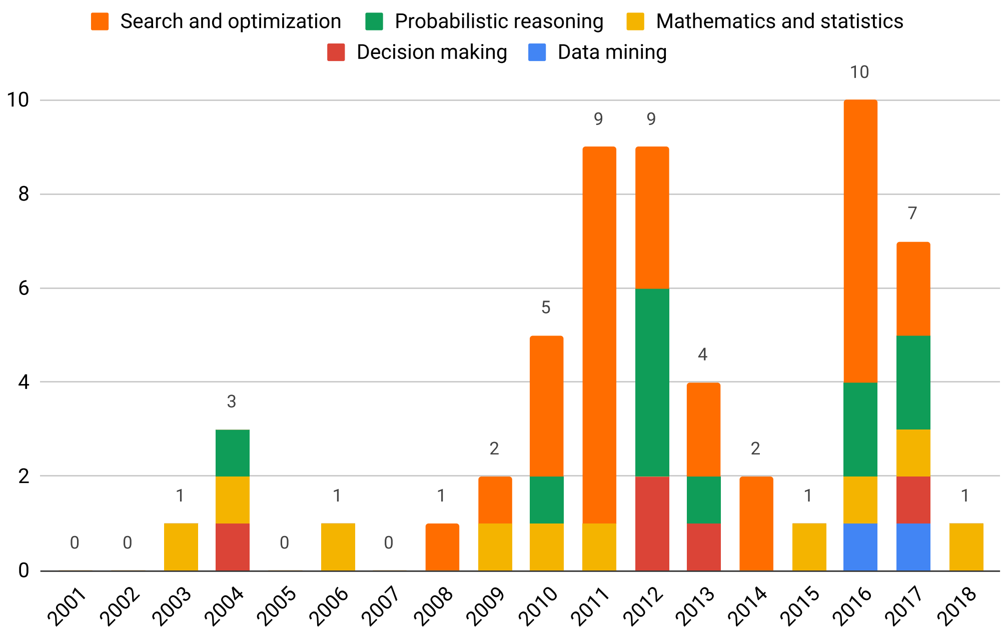


FIGURE 15. Distribution of the technique categories over the years.

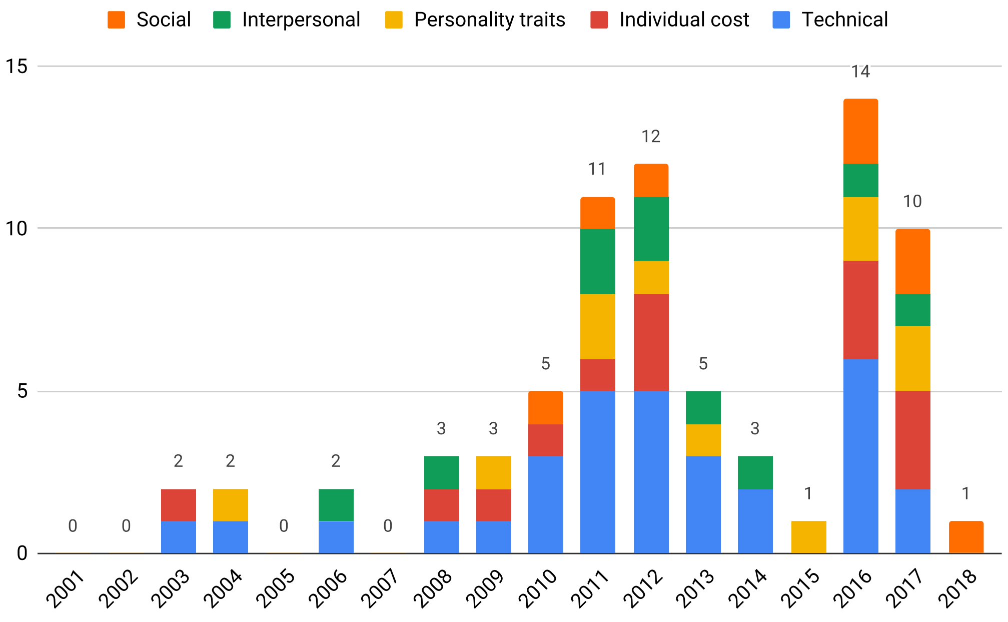


FIGURE 16. Distribution of the attribute types over the years.

or not. We did not identify any study with a level higher than four of automation. We believe this occurs because of the high risks associated with excessive automation for the task of forming software teams. As mentioned before, human resource allocation is a critical factor in a project's success or failure. Given the challenges associated with characterizing the individuals' profiles, projects' requirements, and modeling the relationship between candidate members to predict the team efficiency, the software team formation is a process that highly depends on human knowledge and reasoning. Given our experience and the data collected in this study, we believe that this paradigm will not change, but tools can help humans to make better decisions.

Regarding the research gaps, we identified several points in the proposed solutions, which are summarized in Table 7. As mention before, the team formation problem is NP-hard. However, most of the solutions are validated with toy examples or in simplistic scenarios. In real-world environments, these solutions might not be scalable, i.e., they will probably fail to provide proper support in complex scenarios. For instance, Figure 12 points out that 15% of the solution outputs a *Single team*. A single team selection tool will probably not

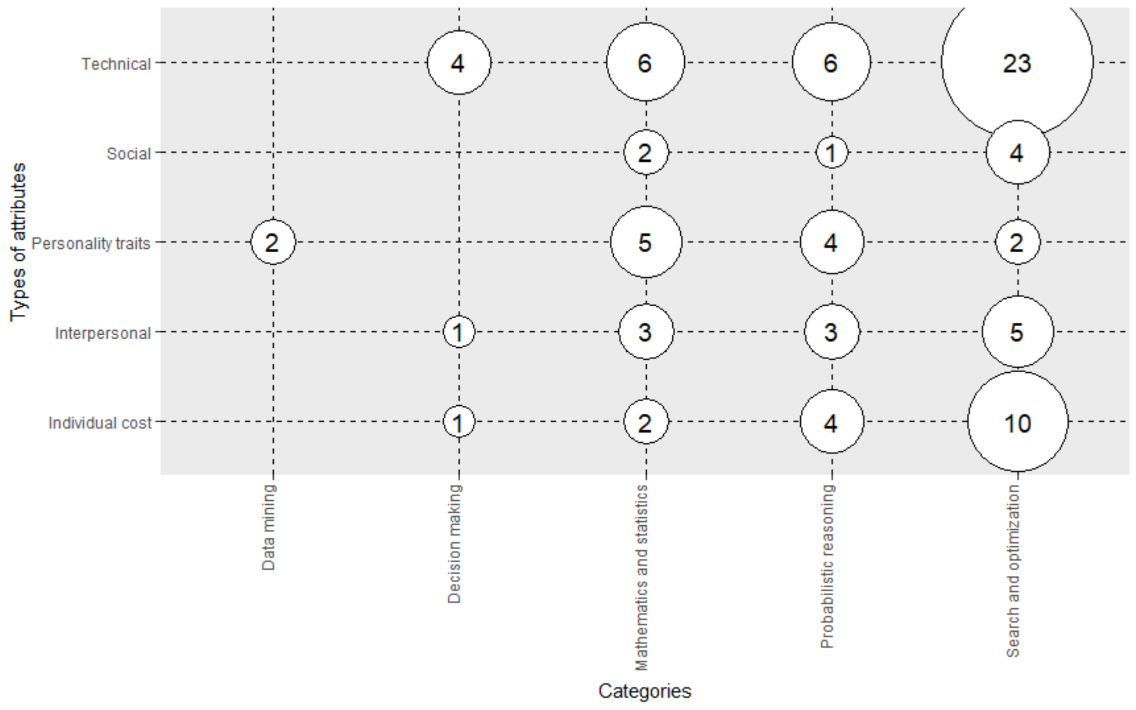


FIGURE 17. Frequency of combination between the technique categories and types of attributes.

TABLE 6. Levels of automation of decision and action. Classification extracted from [33].

Level	Description
10	Computer makes and implements decision if it feels it should, and informs human only if it feels this is warranted.
9	Computer makes and implements decision, and informs human only if it feels this is warranted.
8	Computer makes and implements decisions, and informs human only if asked to.
7	Computer makes and implements decisions but must inform human after the fact.
6	Computer makes decisions but gives the human option to veto before implementation.
5	Computer offers a restricted set of alternatives and suggests one, which it will implement if human approve.
4	Computer offers a restricted set of alternatives and suggests one, but human still makes and implements the final decision.
3	Computer offers a restricted set of alternatives, and human decides which to implement.
2	Computer offers a set of alternatives which the human may ignore when making the decisions.
1	human considers alternatives, makes and implements the decisions.

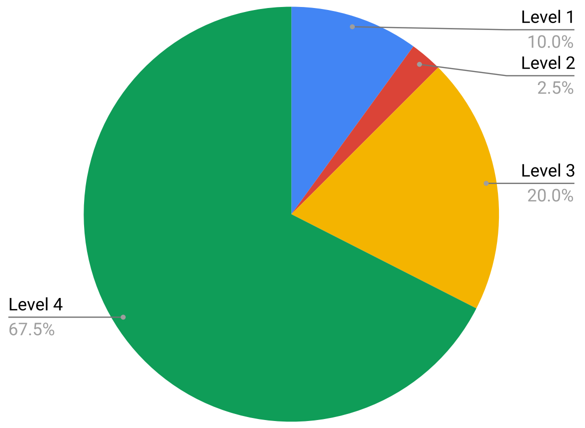


FIGURE 18. Levels of automation identified in the included studies.

TABLE 7. Summary of the identified gaps.

#	Gaps
1	Solutions are validated with toy examples.
2	High level of abstraction and subjectivity in individuals' attributes.
3	Poor assessment of the proposed solutions' results.
4	Lack of solutions and data made available to the community.

be useful for a large company with dozens of projects and hundreds of employees. Similarly, small companies may not need an automated tool to form a single team.

We also identified a gap related to the construction of the individuals' profiles. Most of the studies define a set of attributes to represent the individuals' knowledge and skills to form teams that match projects' requirements. However, these attributes are usually specified with a high level of abstraction. For instance, when quantifying programming language experience, the considered attribute does not specify the languages, it only has a score representing the attribute magnitude. Therefore, it is not possible to know exactly how extensive the individual's experience is.

Moreover, the attributes' scores are usually defined by the project managers, by the individual himself or by his coworkers. This practice results in subjective and error-prone profiles since the data created may not reflect reality. We believe this gap occurs because most solutions focus on the allocation step and do not provide a systematic way to record the individuals' knowledge and skills.

Another gap regards the assessment of the proposed solutions' results. Figure 13 indicates that 7.50% do not provide any assessment and 40% provide expert judgment. A proper assessment is essential to know the extent to which the proposed solution can support the software team formation. Furthermore, human judgment is predisposed to cognitive biases; consequently, it is possible to pose a potential threat in the validity of this type of assessment.

Finally, we identified a gap concerning the lack of solutions and data made available to the community. The identified studies do not provide the necessary means to enable other researchers to use their solutions or replicate their studies, which hinders their evaluation in different contexts. Consequently, there is a lack of studies comparing their results with similar ones available in the literature. Kitchenham *et al.* [34] suggest that whenever a new solution is proposed, it is crucial to compare its results in contrast to existing ones. In particular, these two gaps indicate the area has still not reached an adequate level of maturity.

IV. ORGANIZING THE KNOWLEDGE ON SOFTWARE TEAM FORMATION

A taxonomy is a valuable tool to organize knowledge in an area. It allows the description of terms and their relationships, which is beneficial for both practitioners and researchers [35]. Based on the achieved results, we concluded that the software team formation area is maturing; however, it did not reach an adequate level yet. For instance, during this study, we analyze titles, abstracts, and keywords of the primary studies. As a result, we identified 16 different terms that refer to the research topic, presented in Figure 19. *Team formation* is the most popular term, present in 27, 50% of the studies; however, there is no consensus in the area. This lack of a common terminology makes it difficult for researchers and practitioners to share knowledge, identify gaps in the area, understand the interrelationships between the factors associated with the area, and make decisions [36].

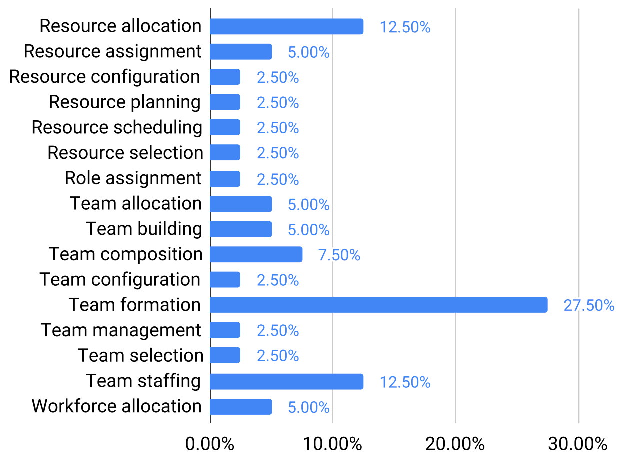


FIGURE 19. Research field's common terms.

To the best of our knowledge, there is no taxonomy for software team formation. Therefore, we decided to propose a taxonomy to organize and communicate the knowledge in this area for the research community and industrial practitioners. Usman *et al.* [35] proposed a method for taxonomy development, which has 4 phases with 13 activities. The development process and the proposed taxonomy are presented next.

A. PHASES AND ACTIVITIES

1) PLANNING

This phase is composed of six activities. First, we defined the Software Engineering knowledge area for the taxonomy. For this purpose, we used the classification of SWEBOK [37] and selected *Software Engineering Management*. Afterward, we determined the objective of our taxonomy, which is to propose a classification scheme that can be used to characterize software team formation. The third activity consists of specifying the subject matter of the classification, which is *Software Team Formation research*. As we mentioned before, team formation is the most used term in the research topic. We then chose a facet-based classification structure because it can provide more than one perspective to view and classify the research area. Facet-based structures are also used in [35], [36], [38]. Additionally, each facet is independent and can have its own classes, which allow the taxonomy to be easily evolved. Next, we selected qualitative type as classification procedure type, since our classification is based on nominal scales. The last activity of this phase is to identify information sources to extract the relevant data. In this study, the source consists of peer-reviewed empirical studies on software team formation published between 2001 and 2018.

2) IDENTIFICATION AND EXTRACTION

This phase has activities for extracting and controlling the terms related to the subject matter. First, we extracted relevant terms from the data collected in this study. The second activity is the terminology control, which consists of removing remove inconsistencies in the extracted data.

3) DESIGN AND CONSTRUCTION

This phase comprises four activities to assist the definition of dimensions, categories and relationships of the taxonomy. First, we defined the top-level dimensions as *Team characteristics*, *Solution*, and *Criteria*. Then, we specified the categories for each of the dimensions. The *Team characteristics* dimension is composed of *development methodology*, *geographical distribution*, *team roles*, and *team size*. The *Solution* dimension incorporates *objective*, *technique category*, *output*, and *level of automation*. The *Criteria* dimension has *attribute type* and *attribute assignment* as categories.

4) VALIDATION

This is the last phase of the taxonomy development method proposed by Usman *et al.* [35]. The purpose of this phase is to demonstrate the working or usefulness of the designed taxonomy. Existing validation approaches for a taxonomy include demonstrating the orthogonality of its categories, benchmarks against existing taxonomies, and by demonstrating its utility to classify existing knowledge [39], [40]. Since there is no existing taxonomy on software team formation, benchmark the proposed taxonomy against existing ones is not possible. Our taxonomy's orthogonal demonstration is ensured by design. The three dimensions (Figure 20) are distinguishable from each other and their categories are mutually exclusive. Concerning the utility demonstration, the taxonomy was created based on the identified studies in this SMS. Theoretically, by performing this SMS, we already found all the relevant studies. Consequently, we cannot demonstrate the taxonomy's utility since it would not be appropriate to apply the same set used to build it. Therefore, we intend to demonstrate taxonomy's utility in future work.

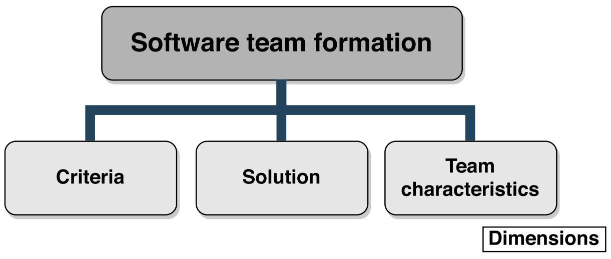


FIGURE 20. **Taxonomy dimensions.**

B. OVERVIEW OF THE PROPOSED TAXONOMY

We organized the identified knowledge of software team formation as a taxonomy. Figure 20 shows the facet-based classification structure and its dimensions: *Team characteristics*, *Solution*, and *Criteria*.

Figure 21 presents the *Criteria* dimension which represents attribute related characteristics (see Section III-C). This dimension is composed of two facets and seven facet values. To preserve the taxonomy's orthogonality, we merged the attribute types (Figure 8) and attribute assignments (Figure 9). The *Attribute type* facet is composed of *Technical*, *Non-technical* (i.e., personality traits, interpersonal skills, social skills, and others), and *Both* (when the solution considers technical and non-technical attributes). The *Attribute assignment* facet comprises *Data-driven* (when the values derived from historical data or similar sources), *Expert judgment* (when specialists are responsible for assigning the values), *Collaborative assessment* (when the individual himself or his coworkers assigned the values), and *Other* (when the previous three facet values do not apply).

The *Solution* dimension (Figure 22) has four facets and 16 facet values. It represents characteristics related to the solutions identified in the literature (see Section III-D). Regarding the *Objective* facet, we merged the objectives

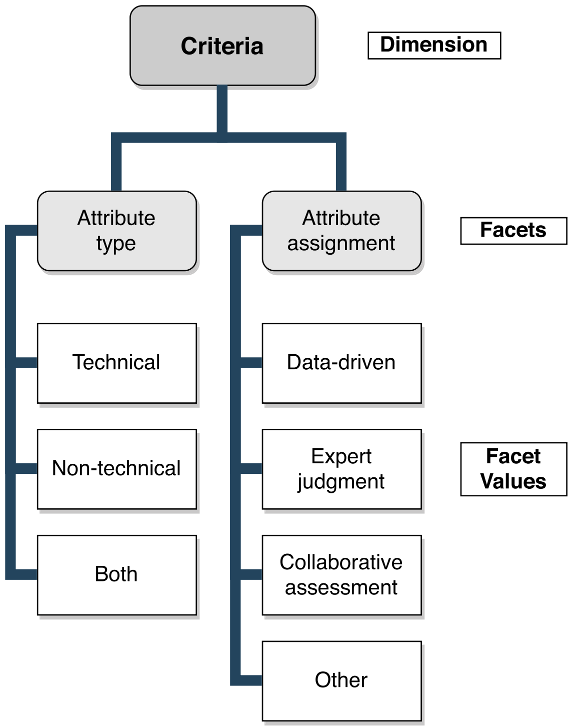


FIGURE 21. **Taxonomy dimension: criteria.**

(Figure 10) *Maximize project's requirements*, *Minimize project's cost* and *Minimize project's delivery* into *Satisfy project's constraint* to the preserve the taxonomy's orthogonality. Similarly, some identified studies use more than one technique in their solutions (Figure 11); thus the *Technique category* facet must represent the main technique of the solution, i.e., the technique applied to optimize the team formation process. The *Output* facet was extracted from the results presented in Figure 12. Finally, the *Level of automation* facet was based on the classification presented by Feldt *et al.* [33].

The *Team characteristics* dimension (Figure 23) includes four facets and 13 facet values. This dimension incorporates the main characteristics to be considered during the team formation process (see Section III-C). The novelty is the *Team size* facet which can be *Small* (3 to 9 members), *Medium* (10 to 30 members), and *Large* (31 or more members). We used the team size classification presented in [41]. The remaining categories were derived from the data collected during this SMS.

V. THREATS TO VALIDITY

As well as other empirical studies, this SMS also has limitations that must be considered for analyzing the potential impact of the validity threats related to its findings. We pointed out the following three main types of validity threats associated with the activities of this SMS.

A. REGARDING PRIMARY STUDIES IDENTIFICATION

In the literature search strategy, we aimed to retrieve as many relevant studies as possible to avoid any potential

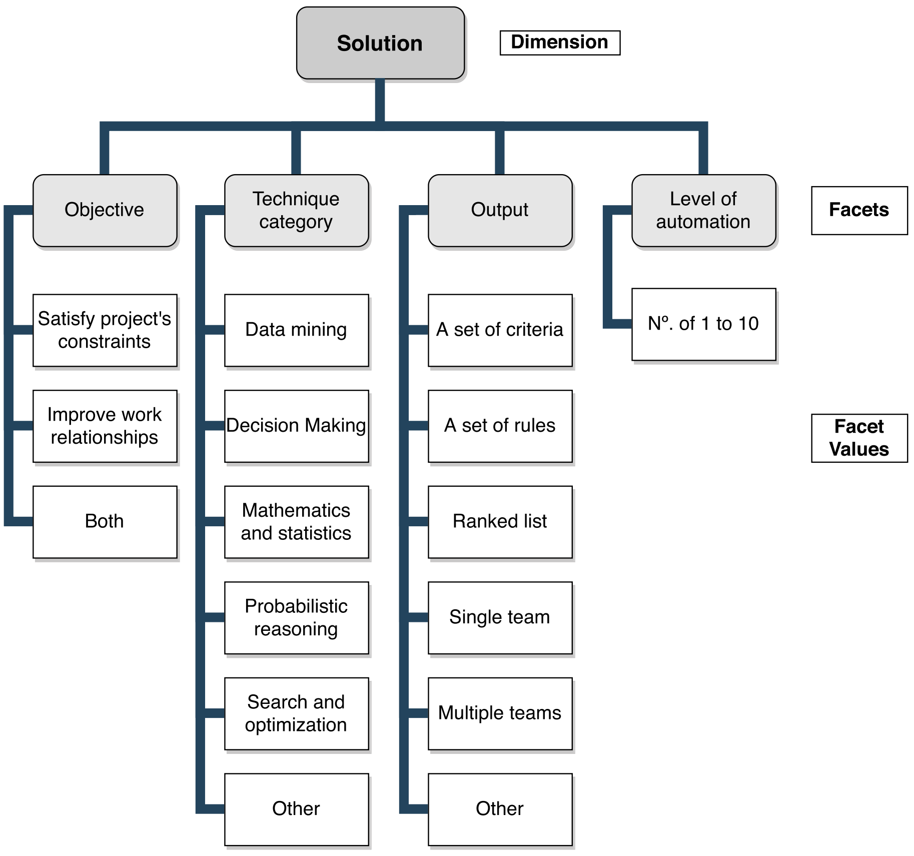


FIGURE 22. Taxonomy dimension: solution.

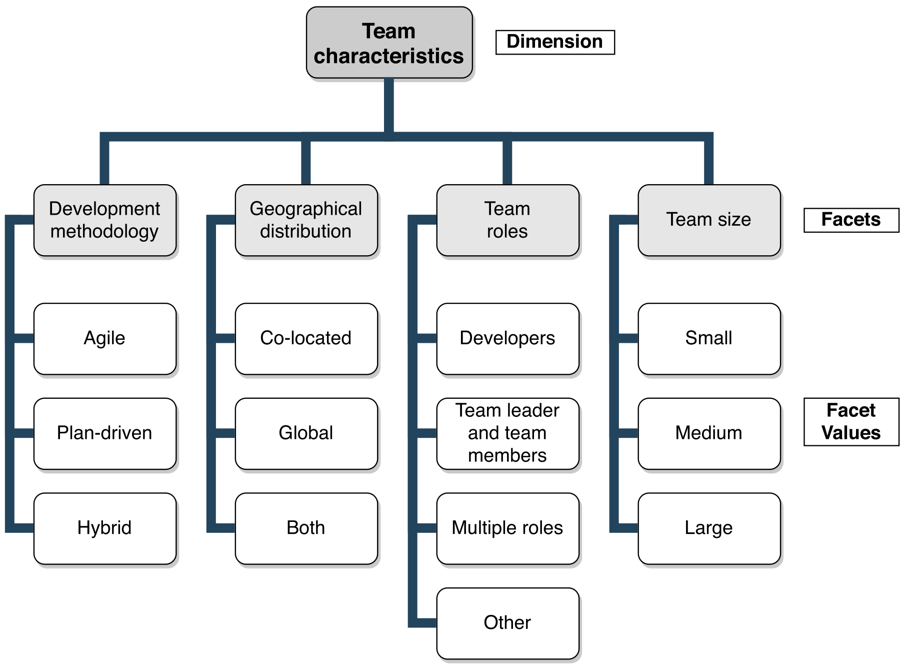


FIGURE 23. Taxonomy dimension: team characteristic.

literature selection bias. We faced a challenge in determining our research scope as the notion of "team formation" means different things to different research communities, including software engineering, artificial intelligence, agile development, and many others. Therefore, to cover them all and avoid bias, we searched the literature based on relevant

TABLE 8. **Basic information of the included papers.**

Paper ID	Study ID	Title	Snowballing phase	Reference
P01	S01	Team Member Selection In Agile	Start Set	[43]
P02	S02	Resolving team selection in agile development using NSGA-II algorithm	Start Set	[44]
P03	S03	Skill-based Team Formation in Software Ecosystems	Start Set	[45]
P04	S04	Applying Fuzzy Technique In Software Team Formation Based On Belbin Team Role	Start Set	[46]
P05	S04	Towards a balanced software team formation based on Belbin team role using fuzzy technique	Start Set	[47]
P06	S05	A rule-based model for software development team composition: Team leader role with personality types and gender classification	Start Set	[7]
P07	S05	Balancing The Personality Of Programmer: Software Development Team Composition	Start Set	[48]
P08	S06	A hybrid approach to solve the agile team allocation problem	Iteration 1	[49]
P09	S07	Capacitated team formation problem on social networks	Iteration 1	[13]
P10	S08	The use of search-based optimization techniques to schedule and staff software projects: An approach and an empirical study	Iteration 1	[11]
P11	S09	Novel approach to multi-functional project team formation	Iteration 1	[50]
P12	S10	Who should work with whom? Building effective software project teams	Iteration 1	[51]
P13	S05	A rule-based approach for discovering effective software team composition	Iteration 1	[52]
P14	S05	Discovering personality types and diversity based on software team roles	Iteration 1	[53]
P15	S11	Experiences in software engineering courses using psychometrics with RAMSET	Iteration 1	[54]
P16	S12	Software developer selection: A holistic approach for an eclectic decision	Iteration 1	[55]
P17	S05	A Set of Rules for Constructing Gender-based Personality types' Composition for Software Programmer	Iteration 1	[56]
P18	S13	Exploitation of social semantic technology for software development team configuration	Iteration 1	[57]
P19	S14	Staffing a software project: A constraint satisfaction and optimization-based approach	Iteration 2	[58]
P20	S14	Staffing a software project: A constraint satisfaction approach	Iteration 2	[59]
P21	S15	Human resource selection for software development projects using Taguchi's parameter design	Iteration 2	[8]
P22	S16	Forming Grouped Teams with Efficient Collaboration in Social Networks	Iteration 2	[60]
P23	S17	A multi-objective genetic algorithm for intelligent software project scheduling and team staffing	Iteration 2	[61]
P24	S18	A novel prototype tool for intelligent software project scheduling and staffing enhanced with personality factors	Iteration 2	[62]

TABLE 8. ***(Continued.)*** **Basic information of the included papers.**

Paper ID	Study ID	Title	Snowballing phase	Reference
P25	S19	Cooperative co-evolutionary optimization of software project staff assignments and job scheduling	Iteration 2	[63]
P26	S20	A fuzzy-genetic decision support system for project team formation	Iteration 2	[64]
P27	S21	A quantitative assessment on team building criteria for software project teams	Iteration 2	[65]
P28	S21	An empirical study on the use of team building criteria in software projects	Iteration 2	[6]
P29	S22	Formal model for assigning human resources to teams in software projects	Iteration 2	[4]
P30	S23	Forming successful extreme programming teams	Iteration 2	[66]
P31	S05	Making programmer suitable for team-leader: Software team composition based on personality types	Iteration 2	[67]
P32	S24	Managing Software Engineering Student Teams Using Pellerin's 4-D System	Iteration 2	[68]
P33	S25	Personalities And Software Development Team Performance, A Psycholinguistic Study	Iteration 2	[69]
P34	S26	Supporting agile team composition: A prototype tool for identifying personality (In) compatibilities	Iteration 2	[70]
P35	S11	Using MatLab's fuzzy logic toolbox to create an application for RAMSET in software engineering courses	Iteration 2	[71]
P36	S27	Resyster: A hybrid recommender system for scrum team roles based on fuzzy and rough sets	Iteration 2	[72]
P37	S18	A Multi-objective Genetic Algorithm for Software Development Team Staffing Based on Personality Types	Iteration 3	[73]
P38	S28	A multi-criteria approach for team recommendation	Iteration 3	[74]
P39	S29	A multi-objective genetic algorithm for software personnel staffing for HCIM solutions	Iteration 3	[75]
P40	S30	Decision model for allocating human resources in information system projects	Iteration 3	[76]
P41	S31	Minimal cost stable workforce allocation in presence of ties	Iteration 3	[77]
P42	S32	Scatter search for trainees to software project requirements stable allocation	Iteration 3	[78]
P43	S33	Skill-based framework for optimal software project selection and resource allocation	Iteration 3	[79]
P44	S34	Staffing open collaborative projects based on the degree of acquaintance	Iteration 3	[80]
P45	S35	A multi-criteria decision making approach for resource allocation in software engineering	Iteration 3	[81]
P46	S36	An ontology-based approach with which to assign human resources to software projects	Iteration 3	[82]
P47	S37	Binary fuzzy goal programming for effective utilization of IT professionals	Iteration 3	[83]
P48	S38	Measuring social networks when forming information system project teams	Iteration 3	[2]
P49	S05	Finding an effective classification technique to develop a software team composition model	Iteration 3	[84]
P50	S39	Value-based multiple software projects scheduling with genetic algorithm	Iteration 4	[85]
P51	S40	Recommending people in developers' collaboration network	Iteration 4	[86]

TABLE 9. **Research data of the included papers.**

Research question	Research result	Research validation	Reference
Method or means of development	Procedure or technique	Blatant assertion	[43]
Method or means of development	Procedure or technique	Example	[44]
Method or means of development	Procedure or technique	Analysis	[45]
Method or means of development	Procedure or technique	Blatant assertion	[46]
Method or means of development	Procedure or technique	Blatant assertion	[47]
Method for analysis or evaluation	Empirical model	Analysis	[7]
Method or means of development	Procedure or technique	Analysis	[48]
Method or means of development	Procedure or technique	Evaluation	[49]
Method or means of development	Procedure or technique	Analysis	[13]
Method for analysis or evaluation	Procedure or technique	Evaluation	[11]
Method or means of development	Procedure or technique	Example	[50]
Design, evaluation, or analysis of a particular instance	Report	Evaluation	[51]
Method or means of development	Empirical model	Analysis	[52]
Method for analysis or evaluation	Specific solution, prototype, ...	Example	[53]
Method or means of development	Procedure or technique	Evaluation	[54]
Method or means of development	Procedure or technique	Example	[55]
Method or means of development	Procedure or technique	Evaluation	[56]
Method or means of development	Procedure or technique	Evaluation	[57]
Method or means of development	Tool or notation	Evaluation	[58]
Method or means of development	Procedure or technique	Example	[59]
Method or means of development	Procedure or technique	Analysis	[8]
Method or means of development	Procedure or technique	Analysis	[60]
Method or means of development	Procedure or technique	Analysis	[61]
Method or means of development	Tool or notation	Evaluation	[62]
Method or means of development	Procedure or technique	Analysis	[63]
Method or means of development	Procedure or technique	Evaluation	[64]
Method for analysis or evaluation	Specific solution, prototype, ...	Analysis	[65]
Method for analysis or evaluation	Empirical model	Analysis	[6]
Method or means of development	Tool or notation	Evaluation	[4]
Method for analysis or evaluation	Specific solution, prototype, ...	Evaluation	[66]
Method or means of development	Procedure or technique	Evaluation	[67]
Method or means of development	Tool or notation	Evaluation	[68]
Method or means of development	Empirical model	Analysis	[69]
Method or means of development	Tool or notation	Evaluation	[70]
Method or means of development	Tool or notation	Evaluation	[71]
Method or means of development	Procedure or technique	Evaluation	[72]
Method or means of development	Procedure or technique	Analysis	[73]
Method or means of development	Empirical model	Analysis	[74]
Method or means of development	Tool or notation	Evaluation	[75]
Method for analysis or evaluation	Qualitative or descriptive model	Analysis	[76]
Method or means of development	Procedure or technique	Evaluation	[77]
Method or means of development	Procedure or technique	Evaluation	[78]
Method or means of development	Tool or notation	Experience	[79]
Method for analysis or evaluation	Procedure or technique	Evaluation	[80]
Method or means of development	Procedure or technique	Example	[81]
Method or means of development	Procedure or technique	Evaluation	[82]
Method or means of development	Procedure or technique	Example	[83]
Method or means of development	Procedure or technique	Evaluation	[2]
Method or means of development	Empirical model	Analysis	[84]
Method or means of development	Procedure or technique	Example	[85]
Method or means of development	Procedure or technique	Evaluation	[86]

TABLE 10. Publication data of the included papers.

Publication type	Publication year	Publication name	Reference
Journal	2016	International Journal Of Science Technology & Management	[43]
Journal	2016	CSI Transactions on ICT	[44]
Workshop	2016	International Workshop on Quality Assurance in Computer Vision and the International Workshop on Digital Eco-Systems	[45]
Journal	2016	Journal of Telecommunication, Electronic and Computer Engineering	[46]
Conference	2016	International Conference On Applied Science And Technology	[47]
Journal	2016	Information and Software Technology	[7]
Journal	2016	Malaysian Journal of Computer Science	[48]
Conference	2012	Congress on Evolutionary Computation	[49]
Conference	2012	International conference on Knowledge discovery and data mining	[13]
Journal	2011	Journal: Software - Practice & Experience	[11]
Journal	2004	International Journal of Project Management	[50]
Magazine	2004	Communications of the ACM - Wireless sensor networks	[51]
Journal	2014	Journal of Information & Communication Technology	[52]
Conference	2013	International Conference on Computing and Informatics	[53]
Conference	2010	Annual conference on Innovation and technology in computer science education	[54]
Journal	2012	International Journal of Computer Applications	[55]
Conference	2015	International Conference on Advanced Data and Information Engineering	[56]
Journal	2010	Institution of Engineering and Technology	[57]
Journal	2008	Computers and Operations Research	[58]
Workshop	2005	International workshop on Economics-driven software engineering research	[59]
Journal	2003	European Journal of Operational Research	[8]
Journal	2016	The Computer Journal	[60]
Journal	2013	Intelligent Decision Technologies	[61]
Conference	2012	International Conference on Tools with Artificial Intelligence	[62]
Conference	2011	International conference on Search based software engineering	[63]
Journal	2010	Applied Soft Computing	[64]
Workshop	2009	Software Engineering Latin American Workshop	[65]
Symposium	2011	International Symposium on Empirical Software Engineering and Measurement	[6]
Journal	2011	Information and Software Technology	[4]
Conference	2006	Agile Conference	[66]
Symposium	2015	International Symposium on Mathematical Sciences and Computing Research	[67]
Journal	2015	Journal of Information Systems Education	[68]
Conference	2016	European Conference on Information Systems	[69]
Workshop	2009	Workshop on Cooperative and Human Aspects of Software Engineering	[70]
Journal	2013	Computer Applications in Engineering Education	[71]
Journal	2012	International Journal of Applied Mathematics and Computer Science	[72]

TABLE 10. ***(Continued.)*** **Publication data of the included papers.**

Publication type	Publication year	Publication name	Reference
Conference	2012	International Conference on Artificial Intelligence Applications and Innovations	[73]
Conference	2017	International Conference on Business Process Management	[74]
Journal	2014	International Journal of Web Portals	[75]
Journal	2013	International Journal of Project Management	[76]
Conference	2016	International Conference on Industrial Engineering and Engineering Management	[77]
Journal	2017	Journal of Heuristics	[78]
Journal	2014	European Journal of Operational Research	[79]
Conference	2013	International Conference on Database Systems for Advanced Applications	[80]
Conference	2010	International Conference on Computer Modelling and Simulation	[81]
Journal	2018	Science of Computer Programming	[82]
Conference	2016	International Conference on Intelligent Computing and Communication	[83]
Journal	2017	Journal of Systems and Software	[2]
Journal	2017	Journal of Software: Evolution and Process	[84]
Conference	2009	International Conference on Software Process	[85]
Conference	2011	Working Conference on Reverse Engineering	[86]

terms and combined them in our search string. While this search strategy and search string composition significantly increase the search work [15], however, it enabled us to find a comprehensive set of relevant studies.

B. REGARDING THE QUALITY OF STUDIES AND DATA EXTRACTION CONSISTENCY

The results and quality of this empirical study are based on the quality of the reviewed studies. In case the quality of the primary studies is low, the claims and supporting evidence can be weak and lack confidence. Therefore, it is vital to (i) minimize the threats regarding the quality of selected studies and (ii) ensure a consistent representation of data extracted from these studies. The ideal scenarios may strictly adhere to the guidelines in [15], [17], however, the quality metric can be subjective based on the goals of this SMS and the consensus among researchers.

C. REGARDING DATA SYNTHESIS AND RESULTS REPORTING

A limited number of researchers and their expertise (i.e., intelligent software engineering) may have an internal bias on the style and results report. The validity threat related to (i) reliability of data synthesis and (ii) results report was mitigated based on discussion and peer review of the extracted data by the researchers. For this purpose, we used a structured template for data synthesis and performed several steps to refine and evaluate the scheme and process. Whereas, we followed the guidelines described in [15], [17] to conduct this empirical study, we had deviations from the ideal approaches based on the requirements and peculiarity of this study. Finally, it is worth noting the taxonomy proposed in this article is based on the primary studies selected for this research. This means the completeness of the proposed taxonomy is directly related to the primary studies included in the data extraction activity. To minimize this threat's effect, we seek covering the literature more broadly by following the guidelines mentioned above.

VI. CONCLUSION

Forming an appropriate team is crucial for a software project's success. As a result, there has been an increasing number of studies in this context. Our SMS identified 51 primary studies, further classified into 40 distinct studies. Most of the proposed solutions (62.50%) approach the given problem as a *Search and optimization* one, which characterizes them into the growing field of Search-based Software Engineering [42]. In this context, the most popular technique is the Genetic Algorithm (20.00%). Other techniques identified are the Greedy Method, Dynamic Programming, Linear Programming, and Backtracking (i.e., Branch and Bound).

Most of the identified solutions (80.00%) consider *Technical* attributes to build the individuals' profiles. However, we noticed a significant number of recent proposals that consider non-technical attributes to form software teams. All of the solutions focus on supporting humans during the team formation process. They provide (semi-)automatic methods to optimize the allocation of individuals into teams to support the final decision.

TABLE 11. Quality points of the included papers.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	Total	Ref.
1	1	0	0	0	0	0	0	0	0	0	2	[43]
1	1	0	1	0	0	0	0	0	0	0	3	[44]
1	1	1	1	1	0	1	1	0	1	1	9	[45]
1	1	0	1	0	0	0	0	0	0	0	3	[46]
1	1	0	1	0	0	0	0	0	0	0	3	[47]
1	1	1	1	1	0	1	1	1	1	1	10	[7]
1	1	1	1	1	0	1	1	0	1	0	8	[48]
1	1	1	1	1	1	1	1	0	1	1	10	[49]
1	1	1	1	1	0	1	1	0	1	0	8	[13]
1	1	1	1	1	0	1	1	1	1	1	10	[11]
1	1	1	1	0	0	0	0	0	1	1	6	[50]
1	1	1	0	1	0	1	0	0	1	0	6	[51]
1	1	1	1	1	0	1	1	0	1	1	9	[52]
1	1	0	0	0	0	0	1	0	1	0	4	[53]
1	1	1	1	1	0	1	0	0	1	1	8	[54]
1	1	0	1	0	0	0	0	0	0	1	4	[55]
1	1	1	0	0	0	1	1	0	1	1	7	[56]
1	1	1	1	1	0	1	1	0	1	1	9	[57]
1	1	1	1	1	1	1	1	1	1	1	11	[58]
1	1	1	1	0	0	0	0	0	1	1	6	[59]
1	1	1	1	1	0	1	1	0	1	1	9	[8]
1	1	1	1	1	0	1	1	0	1	1	9	[60]
1	1	1	1	1	0	1	1	0	1	1	9	[61]
1	1	1	1	1	0	1	1	0	1	1	9	[62]
1	1	1	1	1	0	1	1	1	0	1	9	[63]
1	1	1	1	1	0	0	0	0	1	1	7	[64]
1	1	0	1	0	0	1	1	1	1	0	7	[65]
1	1	1	1	1	0	1	1	1	1	1	10	[6]
1	1	1	1	1	0	1	1	0	1	1	9	[4]
1	1	1	1	1	0	1	1	0	1	1	9	[66]
1	1	1	1	1	0	1	0	0	1	1	8	[67]
1	1	1	1	0	0	1	0	1	0	1	7	[68]
1	1	1	0	1	0	1	1	0	1	1	8	[69]
1	1	1	1	0	0	0	0	0	0	0	4	[70]
1	1	1	0	1	0	1	0	0	0	0	5	[71]
1	1	1	1	1	0	1	1	0	1	1	9	[72]
1	1	1	0	1	1	1	0	1	1	1	9	[73]
1	1	0	0	1	0	1	1	0	1	0	6	[74]
1	1	0	0	1	0	0	0	1	0	1	5	[75]
1	1	0	0	0	0	1	1	0	1	0	5	[76]
1	1	1	0	1	0	0	1	1	0	0	6	[77]
1	1	1	0	1	0	1	0	0	1	1	7	[78]
1	1	1	0	0	1	1	1	0	1	0	7	[79]
1	1	1	1	1	0	1	1	0	1	1	9	[80]
1	1	0	0	0	0	0	0	0	0	0	2	[81]
1	1	1	0	0	0	1	0	0	1	1	6	[82]
1	1	1	0	0	0	0	0	0	0	1	4	[83]
1	1	1	1	1	1	1	1	1	1	1	11	[2]
1	1	1	1	1	0	1	1	1	1	1	10	[84]
1	1	0	1	0	0	0	0	0	1	1	5	[85]
1	1	1	1	1	0	1	0	1	1	1	9	[86]

The main limitations of the identified solutions are that most of them rely on subjective attributes and may lack scalability. We believe removing the subjectivity is not possible, especially, for soft-skills (i.e., non-technical attributes), but it can be reduced. For instance, instead of asking a manager to assess the individual's knowledge and skills concerning technology, information regarding his productivity and delivered artifacts' quality can be retrieved from historical data of projects in which he participated. We believe that reducing the subjectivity would remove bias from the managers and, consequently, reduce the effort of using the proposed solutions. Furthermore, most of the identified solutions might not be suitable for a real-world environment since they used toy examples.

Regarding the studies' methodology, we did not identify solutions available for use, neither the data collected throughout their studies. As a consequence, reproducing the studies becomes unfeasible, and the cost of (i) executing comparative studies against the available tools and (ii) transferring the technology to the industry are high.

From an industry perspective, the results of our study provide knowledge on the key attributes, techniques, and other characteristics to form software teams. Through this study, practitioners can improve their decision-making process while forming a software team. Regarding the researchers, this study identifies essential limitations on the state of the art that must be addressed. Our findings indicate a need to define a gold standard regarding the attributes, metrics, and techniques used to form software teams. We believe that this will be possible if researchers make available the developed tools and data, following the open science initiative.

As future work, we intend to investigate further some of the research directions presented in this research by elaborating on new research questions on software team formation. For example, we could look closely at the techniques that support the various dimensions and facets of the proposed taxonomy. Additionally, we can analyze in-depth, which are the most relevant technical and non-technical attributes for software team formation, providing an especial catalog. Moreover, we intend to continue this empirical study, extending the number of selected studies by query other relevant scientific databases, and by performing new snowballing procedures. We hope, from the increase of the sample of studies, we can reconfirm the main findings of this SMS or even point out new ones.

APPENDIX
DETAILED INFORMATION OF THE INCLUDED PAPERS

See Table 8–11.

REFERENCES

[1] R. T. Futrell, L. I. Shafer, and D. F. Shafer, *Quality Software Project Management*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[2] R. Latorre and J. Suárez, "Measuring social networks when forming information system project teams," *J. Syst. Softw.*, vol. 134, pp. 304–323, Dec. 2017.

[3] R. N. Charette, "Why software fails [software failure," *IEEE Spectr.*, vol. 42, no. 9, pp. 42–49, Sep. 2005.

[4] M. André, M. G. Baldoquín, and S. T. Acuña, "Formal model for assigning human resources to teams in software projects," *Inf. Softw. Technol.*, vol. 53, no. 3, pp. 259–275, Mar. 2011.

[5] S. T. Acuña and N. Juristo, "Assigning people to roles in software projects," *Softw., Pract. Exper.*, vol. 34, no. 7, pp. 675–696, Jun. 2004.

[6] F. Q. B. da Silva, A. C. C. Franca, T. B. Gouveia, C. V. F. Monteiro, E. S. F. Cardozo, and M. Suassuna, "An empirical study on the use of team building criteria in software projects," in *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2011, pp. 58–67.

[7] A. R. Gilal, J. Jaafar, M. Omar, S. Basri, and A. Waqas, "A rule-based model for software development team composition: Team leader role with personality types and gender classification," *Inf. Softw. Technol.*, vol. 74, pp. 105–113, Jun. 2016.

[8] H.-T. Tsai, H. Moskowitz, and L.-H. Lee, "Human resource selection for software development projects using Taguchi's parameter design," *Eur. J. Oper. Res.*, vol. 151, no. 1, pp. 167–180, Nov. 2003.

[9] H. Y. Chiang and B. M. T. Lin, "A decision model for human resource allocation in project management of software development," *IEEE Access*, vol. 8, pp. 38073–38081, 2020.

[10] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA: ACM, 2009, pp. 467–476, doi: 10.1145/1557019.1557074.

[11] M. Di Penta, M. Harman, and G. Antoniol, "The use of search-based optimization techniques to schedule and staff software projects: An approach and an empirical study," *Softw., Pract. Exper.*, vol. 41, no. 5, pp. 495–519, Apr. 2011.

[12] A. Costa, F. Ramos, M. Perkusich, A. Freire, H. Almeida, and A. Perkusich, "A search-based software engineering approach to support multiple team formation for scrum projects," in *Proc. 30th Int. Conf. Softw. Eng. Knowl. Eng.*, San Francisco Bay, CA, USA, Jul. 2018, pp. 473–474.

[13] A. Majumder, S. Datta, and K. V. M. Naidu, "Capacitated team formation problem on social networks," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2012, pp. 1005–1013.

[14] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015.

[15] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng. (EASE)*, New York, NY, USA: ACM, 2014, pp. 38:1–38:10, doi: 10.1145/2601248.2601268.

[16] N. B. Ali, K. Petersen, and C. Wohlin, "A systematic literature review on the industrial use of software process simulation," *J. Syst. Softw.*, vol. 97, pp. 65–85, Nov. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121214001502

[17] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *J. Syst. Softw.*, vol. 80, no. 4, pp. 571–583, Apr. 2007, doi: 10.1016/j.jss.2006.07.009.

[18] M. Staples and M. Niazi, "Experiences using systematic review guidelines," *J. Syst. Softw.*, vol. 80, no. 9, pp. 1425–1437, Sep. 2007, doi: 10.1016/j.jss.2006.09.046.

[19] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Inf. Softw. Technol.*, vol. 50, nos. 9–10, pp. 833–859, Aug. 2008, doi: 10.1016/j.infsof.2008.01.006.

[20] M. Shaw, "Writing good software engineering research papers: Minitutorial," in *Proc. 25th Int. Conf. Softw. Eng. (ICSE)*, Washington, DC, USA: IEEE Computer Society, 2003, pp. 726–736. [Online]. Available: http://dl.acm.org/citation.cfm?id=776816.776925

[21] M. Hirsch, "Moving from a plan driven culture to agile development," in *Proc. 27th Int. Conf. Softw. Eng. (ICSE)*, 2005, p. 38,

[22] N. Ramasubbu, M. Cataldo, R. K. Balan, and J. D. Herbsleb, "Configuring global software teams: A multi-company analysis of project productivity, quality, and profits," in *Proc. 33rd Int. Conf. Softw. Eng. (ICSE)*, 2011, pp. 261–270.

[23] G. M. Olson and J. S. Olson, "Distance matters," *Hum.-Comput. Interact.*, vol. 15, nos. 2–3, pp. 139–178, 2000.

[24] K. Schwaber and J. Sutherland. (2017). *The Scrum Guide: The Definitive Guide to Scrum*. EUA, Wheaton, IL, USA. Accessed: Jun. 20, 2020. [Online]. Available: https://www.scrumguides.org/scrum-guide.html

[25] M. Omar, Z. A. Aljasim, M. Ahmad, F. Baharom, A. Yasin, H. Mohd, and N. M. Darus, "Team formation model of selecting team leader: An analytic hierarchy process (AHP) approach," *ARPN J. Eng. Appl. Sci.*, vol. 10, no. 3, pp. 1060–1067, 2015.

[26] S. M. Henry and K. Todd Stevens, "Using Belbin's leadership role to improve team effectiveness: An empirical investigation," *J. Syst. Softw.*, vol. 44, no. 3, pp. 241–250, Jan. 1999.

[27] S. Faraj and V. Sambamurthy, "Leadership of information systems development projects," *IEEE Trans. Eng. Manag.*, vol. 53, no. 2, pp. 238–249, May 2006.

[28] L. Wallace and M. Keil, "Software project risks and their effect on outcomes," *Commun. ACM*, vol. 47, no. 4, pp. 68–73, Apr. 2004.

[29] A. Aritzeta, S. Swailes, and B. Senior, "Belbin's team role model: Development, validity and applications for team building," *J. Manage. Stud.*, vol. 44, no. 1, pp. 96–118, Jan. 2007.

[30] A. Furnham, "Myers-Briggs type indicator (MBTI)," in *Encyclopedia of Personality and Individual Differences*. Cham, Switzerland: Springer, 2017, pp. 1–4.

[31] G. Antoniol, M. Di Penta, and M. Harman, "Search-based techniques applied to optimization of project planning for a massive maintenance project," in *Proc. 21st IEEE Int. Conf. Softw. Maintenance (ICSM)*, Sep. 2005, pp. 240–249.

[32] F. Ahmed, L. F. Capretz, S. Bouktif, and P. Campbell, "Soft skills and software development: A reflection from the software industry," *J. Inf. Process. Manage.*, vol. 4, no. 3, p. 171, 2013.

[33] R. Feldt, F. G. de Oliveira Neto, and R. Torkar, "Ways of applying artificial intelligence in software engineering," in *Proc. 6th Int. Workshop Realizing Artif. Intell. Synergies Softw. Eng.*, 2018, pp. 35–41.

[34] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case studies for method and tool evaluation," *IEEE Softw.*, vol. 12, no. 4, pp. 52–62, Jul. 1995.

[35] M. Usman, R. Britto, J. Börstler, and E. Mendes, "Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method," *Inf. Softw. Technol.*, vol. 85, pp. 43–59, May 2017.

[36] M. Usman and E. Mendes, "A taxonomy of Web effort predictors," *J. Web Eng.*, vol. 16, nos. 7–8, pp. 541–570, 2017.

[37] P. Bourque and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. Washington, DC, USA: IEEE Computer Society Press, 2014.

[38] M. Usman, J. Börstler, and K. Petersen, "An effort estimation taxonomy for agile software development," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 27, no. 04, pp. 641–674, May 2017.

[39] G. R. Wheaton, "Development of a taxonomy of human performance: A review of classificatory systems relating to tasks and performance," Amer. Inst. Res., Pittsburgh, PA, USA, Tech. Rep. R-68-756, 1968.

[40] D. Šmite, C. Wohlin, Z. Galvina, and R. Prikladnicki, "An empirically based terminology and taxonomy for global software engineering," *Empirical Softw. Eng.*, vol. 19, no. 1, pp. 105–153, Feb. 2014.

[41] L. R. Vijayasarathy and C. W. Butler, "Choice of software development methodologies: Do organizational, project, and team characteristics matter?" *IEEE Softw.*, vol. 33, no. 5, pp. 86–94, Sep. 2016.

[42] M. Harman and B. F. Jones, "Search-based software engineering," *Inf. Softw. Tech.*, vol. 43, no. 14, pp. 833–839, 2001.

[43] A. S. Jat, P. Kohli, and D. Soni, "Team member selection in agile," in *Proc. 4th Int. Conf. Sci., Technol. Manage.*, 2016, pp. 584–588.

[44] A. Arunachalam, N. P. Nagarajan, V. Mohan, M. Reddy, and C. Arumugam, "Resolving team selection in agile development using NSGA-II algorithm," *CSI Trans. ICT*, vol. 4, nos. 2–4, pp. 83–86, Dec. 2016, doi: 10.1007/s40012-016-0105-0.

[45] D. Schall, "Skill-based team formation in software ecosystems," in *Proc. Int. Workshop Qual. Assurance Comput. Vis. Int. Workshop Digit. Eco-Syst.*, 2016, pp. 35–41.

[46] M. Omar, B. Hasan, M. Ahmad, A. Yasin, F. Baharom, H. Mohd, and N. M. Darus, "Applying fuzzy technique in software team formation based on belbin team role," *J. Telecommun., Electron. Comput. Eng.*, vol. 8, no. 8, pp. 109–113, 2016.

[47] M. Omar, B. Hasan, M. Ahmad, A. Yasin, F. Baharom, H. Mohd, and N. M. Darus, "Towards a balanced software team formation based on belbin team role using fuzzy technique," in *Proc. AIP Conf.*, vol. 1761, 2016, Art. no. 020082.

[48] A. R. Gilal, J. Jaafar, M. Omar, S. Basri, and I. Abdul Aziz, "Balancing the personality of programmer: Software development team composition," *Malaysian J. Comput. Sci.*, vol. 29, no. 2, pp. 145–155, Jun. 2016.

[49] R. Britto, P. S. Neto, R. Rabelo, W. Ayala, and T. Soares, "A hybrid approach to solve the agile team allocation problem," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.

[50] T.-L. B. Tseng, C.-C. Huang, H.-W. Chu, and R. R. Gung, "Novel approach to multi-functional project team formation," *Int. J. Project Manage.*, vol. 22, no. 2, pp. 147–159, Feb. 2004.

[51] N. Gorla and Y. W. Lam, "Who should work with whom?: Building effective software project teams," *Commun. ACM*, vol. 47, no. 6, pp. 79–82, Jun. 2004, doi: 10.1145/990680.990684.

[52] A. R. Gilal, M. Omar, and K. I. Sharif, "A rule-based approach for discovering effective software team composition," *J. Inf. Commun. Technol.*, vol. 13, pp. 1–20, Feb. 2014.

[53] A. R. Gilal, M. Omar, and K. I. Sharif, "Discovering personality types and diversity based on software team roles," in *Proc. Int. Conf. Comput. Informat. (ICOCI)*, 2013, pp. 259–264.

[54] L. G. Martínez, G. Licea, A. Rodríguez-Díaz, and J. R. Castro, "Experiences in software engineering courses using psychometrics with RAMSET," in *Proc. 15th Annu. Conf. Innov. Technol. Comput. Sci. Educ. (ITiCSE)*, 2010, pp. 244–248.

[55] S. Kr. Misra and A. Ray, "Software developer selection: A holistic approach for an eclectic decision," *Int. J. Comput. Appl.*, vol. 47, no. 1, pp. 12–18, Jun. 2012.

[56] A. R. Gilal, J. Jaafar, M. Omar, S. Basri, and I. A. Aziz, "A set of rules for constructing gender-based personality types' composition for software programmer," in *Proc. 2nd Int. Conf. Adv. Data Inf. Eng.*, 2015, pp. 363–374.

[57] R. Valencia-Garcia, F. Garcia-Sanchez, D. Castellanos-Nieves, J. T. Fernández-Breis, and A. Toval, "Exploitation of social semantic technology for software development team configuration," *IET Softw.*, vol. 4, no. 6, pp. 373–385, 2010.

[58] A. Barreto, M. D. O. Barros, and C. M. L. Werner, "Staffing a software project: A constraint satisfaction and optimization-based approach," *Comput. Oper. Res.*, vol. 35, no. 10, pp. 3073–3089, Oct. 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054807000226

[59] A. Barreto, M. Barros, and C. Werner, "Staffing a software project: A constraint satisfaction approach," in *Proc. ACM SIGSOFT Softw. Eng. Notes*, vol. 30, 2005, pp. 1–5.

[60] J. Huang, Z. Lv, Y. Zhou, H. Li, H. Sun, and X. Jia, "Forming grouped teams with efficient collaboration in social networks," *Comput. J.*, vol. 60, pp. 1–16, Nov. 2016.

[61] C. Stylianou and A. S. Andreou, "A multi-objective genetic algorithm for intelligent software project scheduling and team staffing," *Intell. Decis. Technol.*, vol. 7, no. 1, pp. 59–80, Jan. 2013.

[62] C. Stylianou, S. Gerasimou, and A. S. Andreou, "A novel prototype tool for intelligent software project scheduling and staffing enhanced with personality factors," in *Proc. IEEE 24th Int. Conf. Tools with Artif. Intell.*, vol. 1, Nov. 2012, pp. 277–284.

[63] J. Ren, M. Harman, and M. Di Penta, "Cooperative co-evolutionary optimization of software project staff assignments and job scheduling," in *Proc. Int. Symb. Search Based Softw. Eng.*, 2011, pp. 127–141.

[64] D. Strnad and N. Guid, "A fuzzy-genetic decision support system for project team formation," *Appl. Soft Comput.*, vol. 10, no. 4, pp. 1178–1187, Sep. 2010.

[65] A. C. C. França, E. F. Lucena, and F. Q. da Silva, "A quantitative assessment on team building criteria for software project teams," in *Proc. 6th Exp. Softw. Eng. Latin Amer. Workshop (ESELAW)*, 2009, p. 12.

[66] A. Gray, A. Jackson, I. Stamouli, and S. L. Tsang, "Forming successful extreme programming teams," in *Proc. AGILE*, 2006, p. 10.

[67] A. R. Gilal, J. Jaafar, S. Basri, M. Omar, and M. Z. Tunio, "Making programmer suitable for team-leader: Software team composition based on personality types," in *Proc. Int. Symp. Math. Sci. Comput. Res. (iSMSC)*, May 2015, pp. 78–82.

[68] M. Doman, A. Besmer, and A. Olsen, "Managing software engineering student teams using Pellerin's 4-D system," *J. Inf. Syst. Educ.*, vol. 26, no. 4, p. 257, 2015.

[69] M. Farhangian, M. Purvis, M. A. Purvis, and B. T. R. Savarimuthu, "Personalities and software development team performance, a psycholinguistic study," in *Proc. 24th Eur. Conf. Inf. Syst.*, 2016, pp. 1–15.

[70] S. Licorish, A. Philpott, and S. G. MacDonell, "Supporting agile team composition: A prototype tool for identifying personality (In)compatibilities," in *Proc. ICSE Workshop Cooperat. Hum. Aspects Softw. Eng.*, 2009, pp. 66–73.

[71] L. G. Martínez, G. Licea, A. Rodríguez, J. R. Castro, and O. Castillo, "Using MATLAB's fuzzy logic toolbox to create an application for RAMSET in software engineering courses," *Comput. Appl. Eng. Educ.*, vol. 21, no. 4, pp. 596–605, Dec. 2013.

[72] R. Colomo-Palacios, I. González-Carrasco, J. L. López-Cuadrado, and Á. García-Crespo, "ReSySTER: A hybrid recommender system for scrum team roles based on fuzzy and rough sets," *Int. J. Appl. Math. Comput. Sci.*, vol. 22, no. 4, pp. 801–816, Dec. 2012.

[73] C. Stylianou and A. S. Andreou, "A multi-objective genetic algorithm for software development team staffing based on personality types," in *Proc. IFIP Int. Conf. Artif. Intell. Appl. Innov.* Cham, Switzerland: Springer, 2012, pp. 37–47.

[74] M. Arias, J. Munoz-Gama, and M. Sepúlveda, "A multi-criteria approach for team recommendation," in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2016, pp. 384–396.

[75] E. Jiménez-Domingo, R. Colomo-Palacios, and J. M. Gómez-Berbís, "A multi-objective genetic algorithm for software personnel staffing for HCIM solutions," *Int. J. Web Portals*, vol. 6, no. 2, pp. 26–41, Apr. 2014.

[76] L. C. e Silva and A. P. C. S. Costa, "Decision model for allocating human resources in information system projects," *Int. J. Project Manage.*, vol. 31, no. 1, pp. 100–108, Jan. 2013.

[77] M. S. Gharote, R. J. Patil, and S. P. Lodha, "Minimal cost stable workforce allocation in presence of ties," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage. (IEEM)*, Dec. 2016, pp. 1146–1150.

[78] M. Gharote, R. Patil, and S. Lodha, "Scatter search for trainees to software project requirements stable allocation," *J. Heuristics*, vol. 23, no. 4, pp. 257–283, Aug. 2017.

[79] F. A. Zaraket, M. Olleik, and A. A. Yassine, "Skill-based framework for optimal software project selection and resource allocation," *Eur. J. Oper. Res.*, vol. 234, no. 1, pp. 308–318, Apr. 2014.

[80] M. Y. Allaho, W.-C. Lee, and D.-N. Yang, "Staffing open collaborative projects based on the degree of acquaintance," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Cham, Switzerland: Springer, 2013, pp. 385–400.

[81] C. E. Otero, L. D. Otero, I. Weissberger, and A. Qureshi, "A multi-criteria decision making approach for resource allocation in software engineering," in *Proc. 12th Int. Conf. Comput. Modeling Simulation*, 2010, pp. 137–141.

[82] M. A. Paredes-Valverde, M. D. P. Salas-Zárate, R. Colomo-Palacios, J. M. Gómez-Berbís, and R. Valencia-García, "An ontology-based approach with which to assign human resources to software projects," *Sci. Comput. Program.*, vol. 156, pp. 90–103, May 2018.

[83] R. Jana, M. Sanyal, and S. Chakrabarti, "Binary fuzzy goal programming for effective utilization of it professionals," in *Proc. 1st Int. Conf. Intell. Comput. Commun.* Cham, Switzerland: Springer, 2017, pp. 395–405.

[84] A. R. Gilal, J. Jaafar, L. F. Capretz, M. Omar, S. Basri, and I. A. Aziz, "Finding an effective classification technique to develop a software team composition model," *J. Softw., Evol. Process*, vol. 30, no. 1, p. e1920, Jan. 2018.

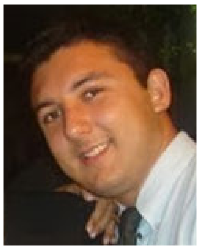
[85] J. Xiao, Q. Wang, M. Li, Q. Yang, L. Xie, and D. Liu, "Value-based multiple software projects scheduling with genetic algorithm," in *Proc. Int. Conf. Softw. Process*. Cham, Switzerland: Springer, 2009, pp. 50–62.

[86] D. Surian, N. Liu, D. Lo, H. Tong, E.-P. Lim, and C. Faloutsos, "Recommending people in developers' collaboration network," in *Proc. 18th Work. Conf. Reverse Eng.*, Oct. 2011, pp. 379–388.

ALEXANDRE COSTA received the M.Sc. and Ph.D. degrees in computer science from the Federal University of Campina Grande, Campina Grande, Brazil, in 2014 and 2019, respectively. He has been a Professor with the Federal Institute of Paraíba, since 2020. Further, he is also a Researcher with the Intelligent Software Engineering Research Group, VIRTUS, which is a Research, Development, and Innovation Center in Information Technology. His current research interest includes artificial intelligence applied to software engineering to solve complex problems. In the software engineering field, his research interests include software project management, agile software development, resource allocation focused on team formation for software development, and others.

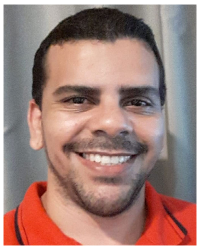
FELIPE RAMOS received the M.Sc. and Ph.D. degrees in computer science from the Federal University of Campina Grande, Campina Grande, Brazil, in 2014 and 2019, respectively. He is currently a Professor with the Federal Institute of Paraíba. Further, he is also a member of the Intelligent Software Engineering Research Group, VIRTUS, which is a Research, Development, and Innovation Center in Information Technology. His current research interest includes artificial intelligence applied to software engineering to solve complex problems. His main research interests include agile software development and requirement engineering focused on supporting the elicitation of non-functional requirements on scrum-based projects.

MIRKO PERKUSICH is currently pursuing the Ph.D. degree in computer science with the Federal University of Campina Grande (UFCG). He is also the Research Manager at Virtus, where he is leading the Intelligent Software Engineering Research Group. His current research interest includes applying intelligent techniques, including recommender systems, to solve complex software engineering problems.



EMANUEL DANTAS received the M.Sc. degree in computer science from the Federal University of Ceará, Fortaleza, Brazil, in 2014. He is currently pursuing the Ph.D. degree with the Federal University of Campina Grande, Campina Grande, Brazil. He has been a Professor with the Federal Institute of Paraíba, since 2015. Further, he is also a Researcher with the Intelligent Software Engineering Research Group, VIRTUS, which is a Research, Development, and Innovation Center in Information Technology. His current research interest includes artificial intelligence applied to software engineering to solve complex problems. In the software engineering field, his research interests include software project management, agile software development, effort estimation, risk management, and others.

EDNALDO DILORENZO received the M.Sc. degree in computer science from the Federal University of Pernambuco, Brazil, in 2012. He is currently pursuing the Ph.D. degree with the Federal University of Campina Grande. He is also a Professor with the Federal Institute for Education, Science, and Technology of Paraíba (IFPB) and a Researcher with the Intelligent Software Engineering Group (ISE/VIRTUS). His research interest includes the application of intelligent techniques to software engineering problems.



FERDINANDY CHAGAS is currently pursuing the Ph.D. degree in computer science with the Federal University of Campina Grande, Brazil. He is also an Assistant Professor with the Federal Rural University of Semi-Arid Region, Brazil. He is a member of the Intelligent Software Engineering (ISE) Research Group, Federal University of Campina Grande. His main research interests include technical debt, software quality, empirical software engineering, ontologies, and multiagent systems.

ANDRÉ MEIRELES is currently pursuing the Ph.D. degree in computer science with the Federal University of Campina Grande, Campina Grande, Brazil. He is also a Professor with the Federal University of Ceará. He is a Researcher at the VIRTUS Research Center. His main research interests include software engineering, smart engineering, software architecture, and edge computing.

DANYLLO ALBUQUERQUE received the M.Sc. degree in informatics from the Federal University of Paraíba, Brazil, in 2013. He is currently pursuing the Ph.D. degree in computer science with the Federal University of Campina Grande. He has been a Professor with the Federal Institute for Education, Science, and Technology of Paraiba (IFPB), since 2020. He is also a Systems Analyst with the Federal University of Campina Grande. Further, he is also a member of the Intelligent Software Engineering Research Group, VIRTUS, which is a Research, Development, and Innovation Center in Information Technology. His current research interests include software quality, technical debt, software architecture, artificial intelligence, and intelligent techniques applied to solve complex software engineering problems.

LUIZ SILVA received the M.Sc. degree in computer science from the Federal University of Campina Grande, Brazil, in 2019. He is currently a Researcher with the Laboratory of Instrumentation and Control (LIEC). His research interests include machine learning and application of intelligent techniques to solve software engineering problems.

HYGGO ALMEIDA received the M.Sc. degree in computer science and the Ph.D. degree in electrical engineering from the Federal University of Campina Grande (UFCG), in 2004 and 2007, respectively. He has been a Professor with the Computer and Systems Department, UFCG, since 2006. He is currently the Head of the Intelligent Software Engineering Group and the Founder and the Director of operations at the VIRTUS Innovation Center (VIRTUS/UFCG). He is a Researcher of the Embedded and Pervasive Computing Laboratory (Embedded/UFCG). He is also the Executive Director of the EMBRAPII Unit at CEEI/UFCG, with more than 150 RDI projects developed in cooperation with industrial companies within the area of information, communication, and automation technologies. He has over 15 years of teaching experience in the university as well as training courses for industry in the context of software engineering. He has more than 200 articles published and 37 master thesis and 13 doctoral dissertations advised. His current research interest includes applying intelligent techniques to solve complex software engineering problems.

ANGELO PERKUSICH received the master's and Ph.D. degrees in electrical engineering from the Federal University of Paraíba, in 1987 and 1994, respectively. He has been a Professor with the Electrical Engineering Department (DEE), Federal University of Campina Grande (UFCG), since 2002. He was a Visiting Researcher with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA, from 1992 to 1993, where he developed research activities on software engineering and formal methods. He is currently the Principal Investigator of research projects financed by public institutions such as FINEP (Brazilian Agency for Research and Studies) and CNPq (Brazilian National Research Council), as well as private companies. He is the Founder and the Director of VIRTUS Innovation Center and the Embedded and Pervasive Computing Laboratory. His focus on research projects is on formal methods, embedded systems, mobile pervasive and ubiquitous computing, and software engineering. He has over 30 years of teaching experience in the university as well as training courses for industry in the context of software for real-time systems, software engineering, embedded systems, computer networks, and formal methods. He has more than 300 articles published and 80 master thesis and 21 doctoral dissertations advised. His main research interests include embedded systems, software engineering, mobile pervasive computing, and formal methods.