www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 15(5), Dec 2024, 237-253. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2024.v15i5.584

Team Composition in Software Engineering Development Phases: A systematic literature review

Nailea Isabel Rosado Castillo[1], Antonio Armando Aguileta Güemez[1], Raúl Antonio Aguilar Vera[1]

[1] Universidad Autónoma de Yucatán, Mérida, Yucatán, México.
A15001585@alumnos.uady.mx, aaguilet@correo.uady.mx, avera@correo.uady.mx

Abstract. The software industry is increasingly competitive and demands rapid adaptation and reduced delivery times for success. Strategic team composition emerges as a fundamental path to contribute to the success of software products and companies. Due to the importance of team composition in software engineering and the existence of various methods for building these teams in the literature, it is essential to understand the current trends and areas for advances in the state of the art on this topic.
Although the literature has extensively examined team composition in software engineering, there is still a significant knowledge gap, particularly regarding the composition throughout various development phases. To address this gap, we conducted a systematic literature review (SLR) focusing on teams built for specific development phases. Our analysis encompasses team composing methods, metrics, size, member characteristics, required skills, and experimental settings.

Keywords: Team Composition, Team Building, Software Engineering, systematic literature review, Software Development Teams.

Article Info
Received May 10,2024.
Accepted Nov 20, 2024.

1 Introduction

The software industry places a higher value on human capital than physical capital, thus highlighting the importance of investing resources in recruiting highly competent personnel [1]. Team composition becomes a crucial process to contribute to the company's success and distinguish itself from the competence in a highly competitive market [2], [3].

In the field of team composition in software engineering, numerous studies have been conducted to experiment with different methodologies for building software products. One of these methodologies is Agile [4], as evidenced by secondary literature studies. For example, Lenberg P. et al. [5] conducted a literature review on behaviour-based software engineering, an area proposed by them, their work focused on finding articles on building software development teams using Agile methodologies. Similarly, Aryanee D. et al. [6] investigated the state of the art of team composition in agile methodologies. Another of the methodologies studied (for software construction, in the context of team composition), is the Waterfall Software Development Life Cycle [7]; as seen in secondary studies. For example, Restrepo L. et al. [8], conducted a literature review on effective team building, focusing on role assignment and highlighting the life cycle model used, whether agile, traditional [9] or hybrid [10].

On the other hand, some researchers have focused on team building for software development phases rather than considering software development as a whole, as described in the aforementioned methodologies for software development. For instance, Aguilar R. et al. [11] experimented in the requirements phase in an educational context, where teams were formed according to Belbin's roles to build the requirements document. Similarly, Aguileta A. et al. [12] studied the implementation phase in an educational context, also using Belbin's roles. The researchers instructed the teams to create software components (requirements and code) to compare the quality.

Despite the relevance of this topic and as demonstrated above, several models for team building have been suggested, to our knowledge, no systematic literature review (SLR) has been carried out that has identified patterns and potential areas of improvement on team composition for different phases of software development, using the Waterfall Model as a reference.

Therefore, this work will conduct a systematic literature review in team composition in software engineering, through the distinct phases of software development, based on the waterfall model. We selected the Waterfall Model since it shares characteristics with other life cycle models [13], such as spiral [14]and iterative [14]. In addition, it has been used as a basis for the creation of other models [15]. To avoid biases towards any particular methodology and enable its application in any methodology or phase required. Particularly, this systematic literature review examines the team composing methods, metrics used to measure the performance of the method, software development team size, characteristics of team members, the necessary skill sets for team members (hard and soft skills), as well as the experimental settings employed.

The paper is structured as follows: Firstly, the research method is described, followed by the planning and execution process. Finally, the results and conclusions of the research are presented.

2 Research Method

To analyse and review the existing literature on team composition in software engineering, this work conducted a systematic literature review (SLR) [16]. The SLR can consist of three distinct phases [17]. The Planning Phase established the objective, research questions, search string, inclusion and exclusion criteria, and information sources. The Execution Phase involves the search and selection of relevant papers, using the search string and applying the previously defined inclusion and exclusion criteria. In the Reporting Phase, the results derived from the analysis of the relevant articles obtained in the previous phases are presented.

3 Planning

3.1. Systematic Literature Review General Objective

The main objective of this systematic literature review is to identify literature on team composition in software engineering, focusing on the various stages of software development according to the Waterfall Model. The analysis covers team composing methods, metrics used to measure the performance of the method, software development team size, characteristics of team members, the necessary skill sets for team members (hard and soft skills), as well as the experimental settings employed.

3.2. Research Question Definition

The research questions will address the overall objective of this systematic literature review. To answer these questions, the articles were classified into two groups: those that focus on building teams for a specific phase of development, classified according to the phases defined by Bassil in [5], and those that address the complete construction of a software product without reference to specific phases, classified under the category of "Software Development." From this point on, we will refer to "development phases," instead of using the term "phases defined by Bassil."

RQ1 What are the similarities, and differences of commonly used methods in software engineering for team composition by development phases?
Criteria considered for analysis: description, method of team building, number of team members, perspective, metrics, experimental subjects, scope, and characteristics considered.

RQ2 What are the experimental settings commonly used by development phases for team composition in the field of software engineering?
Criteria considered for analysis: scope, setting (education, industrial or N/A).

RQ3 What are the characteristics of team members by development phases for team composition in the field of software engineering?
Criteria considered for analysis: method, scope, seven categories: age range, country of origin, Gender, level of expertise, role performed, previous experience, language.

RQ4 What are the hard and soft skills of the team members by development phases for team composition in the field of software engineering?
Criteria considered for analysis: hard and soft skills considered.

RQ5 What are the commonly used team sizes by development phases for team composition in the field of software engineering?
Criteria considered for analysis: number of people considered to form a team.

3.3. Identification and Selection of Sources

The search engine of choice for the implementation phase was Scopus [18]. Scopus provides a comprehensive search of academic literature [19], enabling the compilation of the largest number of research articles available.

3.4. Inclusion and Exclusion Criteria

We defined the inclusion and exclusion criteria to select research articles relevant to this study. For more detailed information, see Table 1.

Table 1. Description of inclusion and exclusion criteria

Inclusion Criteria (IC)	Exclusion Criteria (EC)
IC1. Research articles in the English language	EC1. Research articles published in conference reviews
IC2. Research articles that fall within the time window 2000-2023	EC2. Books or book chapters
IC3. Research articles related to the composition of equipment in the field of software engineering	EC3. Literature reviews or systematic mapping studies

3.5. Search String

The purpose of the search string is to collect relevant research articles for this systematic literature review in the Scopus database. According to the Scopus search engine, we configured the search string. For more specific details, please refer to Table 2.

Table 2. Description of the search string

(TITLE-ABS-KEY (("team composition" OR "team building" OR "building teams" OR "composing teams") AND ("requirements specification" OR "requirement elicitation" OR "analysis" OR "design" OR "development" OR "coding" OR "test" OR "testing" OR "implementation" OR "deployment" OR "operation" OR "maintenance") AND ("software engineer" OR "software engineering")

4 Execution

In June 2023, we conducted a search for academic literature in Scopus using the search string defined above and covering the period from 2000 to June 2023. The search produced a total of ninety-eight research articles. Subsequently, we performed an exhaustive analysis of the obtained research articles. We identified thirty-three research articles by applying the inclusion criteria. After applying the exclusion criteria, for this systematic literature review were considered relevant twenty-seven articles.

5 Reporting and Discussion

RQ1 What are the similarities, and differences of commonly used methods in software engineering for team composition by development phases?

A previous study [20] (see Figure 1 and Figure 2 in the Appendix for further illustration.) analysed the distribution of team composition methods employed to build teams by development phase from a quantitative point of view: Constructive Cost Model (COCOMO) [21] employed in one paper [22], Myers Briggs Type Indicator (MBTI) [23] employed in nine papers across all phases ([24], [25], [26], [27], [28], [29], [30], [31], [32]), "Experimental Team-Building Program" [33] employed in one paper [33], Big Five Model [34] employed in two papers across all phases ([35], [36]).

The Decision Tree [37] was used in three papers across all phases ([24], [28], [32]). The Johnson Algorithm [38] was used in three papers across all phases ([25], [29], [32]). The Genetic Algorithm [39] was used in four papers across all phases ([25], [30], [32], [40]), Belbin's Roles [41] were employed in two papers ([42], [43]) and Keirsey's Personality Type [44] employed in one paper [43]. While IBM Watson Personality Insights [45], Twitpersonality [46], LIWC (Linguistic Inquiry and Word Count) [47], and Personality Recognizer [48] were employed in one paper [36].

Other methods were employed for team composition across all phases such as Team Roles, Gender, and the Umbrella Traversal Model [49], used in four, five, and one paper ([29], [27],[28], [30], [32],[27],[28],[29],[30], and [49], respectively. Conversely, algorithms like principal component regression [50] were employed in one paper [51], logistic regression [52] was employed in one paper [28], shortest path [53] was used in one paper [40], rule-based model [54] in one paper [29], Naïve Bayes Classification [55] was employed in one paper [25], and two papers ([25], [30]) standard voting [56] and voting with object tracking [56].

Similarly, proposed methods like the "Role Assignment Methodology for Software Engineering Teams" were employed in one paper [26]. "Saicho" was employed in one paper [57], and "DELPHI" [58] was also employed in one paper. Therefore, this question will address a deeper analysis, covering aspects such as identifying similarities and differences between methods employed by research papers.

In the analysis of software development phases, specifically in the context of team building, let us start with the requirements phase. The methods employed in this phase were MBTI and COCOMO. The MBTI is a model for identifying personality types based on basic preferences. Personality types include Extraverted, Introverted, Sensing, Intuition, Thinking, Feeling, Judging, and Perceiving. On the other hand, the COCOMO is a model for estimating effort through a series of steps (1. size, 2. effort, 3. duration and 4. cost estimation), where the project size is measured in thousands of delivered lines of code (KDL). For instance, Iqbal M. et al. [59] utilised MBTI to forecast the personality traits of requirements elicitation team members, based on job descriptions for requirements engineers, the study mapped the desired personality traits of a requirements analyst onto the personality traits defined in the MBTI. In contrast, Gulzar W. [22] used COCOMO to determine the optimal size for each phase of software development. The calculation was based on effort and duration estimates for each phase, including the requirements phase.

However, the two studies differ in the number of team members. While Iqbal M. et al. [59] did not mention the number of team members required, Gulzar W. [22] calculated a requirements team of three people. Similarly, Gulzar W. [22] built the requirements team by calculating the necessary number of team members based on the effort and duration estimate of the requirements phase. In contrast, Iqbal M. et al. [59] considered both the personality of the team members and their alignment with the job descriptions. Finally, both researchers proposed models for building teams without using metrics to assess team performance requirements.

Iqbal M. et al. [59] and Gulzar W. [22] have differing perspectives on team construction during the requirements phase. However, these studies only share the characteristic of not using experimental subjects and therefore lack evaluation metrics for the formed teams. It is suggested that there is potential for employing and adapting the proposed models in various contexts to determine their effectiveness in real-world scenarios. In the context of the Design Phase, two different methods were employed without a tendency towards one [20]: COCOMO, the same procedure in the requirements phase applies to the design phase "Experimental Team-Building Program," a proposed method by Hunter K. et al. [33]. The "Experimental Team-Building Program" is a method where team members were randomly assigned to teams, aiming to guide these teams through the initial forming and storming stages of team development [60], and consisted of three components: a fifty-minute lecture

covering fundamental teamwork concepts, a workshop, and a series of follow-up activities over the next three classes. The workshop itself included three activities: Socialization, Team Charter Activity, and Team Initiative: Big Business.

The contrast between these two methods starts with their approach. Gulzar W. [22] used the COCOMO model to determine the necessary number of members based on effort and duration estimates during the design phase. In contrast, Hunter K. et al. [33] aimed to establish whether integrating a team-building programme enhanced course outputs, measured by student development in teamwork knowledge, attitudes towards teamwork, and the quality of the design product. Regarding the optimal team size in the design phase, Gulzar W. [22] found that five-point-one people were required, whereas Hunter K. et al. [33] assigned subjects to teams of eight.

The methods used to construct the teams also differed. Gulzar W. [22] considered both the effort and duration of the design phase when forming the design team, while Hunter K. et al. [33] chose to form teams randomly. The evaluation of team performance was approached differently by Gulzar W. [22] and Hunter K. et al. [33]. Gulzar W. [22] proposed a method to determine the required number of team members for each development phase but did not use any measures to evaluate the performance of the teams formed, while Hunter K. et al. measured the improvement of course outputs. Gulzar W. [22] and Hunter K. et al. [33] have different perspectives on team building in the design phase. The authors select the number of members and evaluation methods based on their research focus. These differences between research works enrich the landscape in different sectors of software engineering. This highlights the importance of considering different contexts in which the design phase takes place. It also opens new fields for adapting and improving design team-building methods.

During the Implementation phase several methods were employed, to classify them we propose four categories: personality models, algorithms, tools, and others. The personality model category involves methods that consider the personalities of team members. Likewise, the algorithm category includes methods based on algorithms for building teams, while the tools category includes approaches that use external tools or methods for building teams. Lastly, the other category refers to methods of building teams that do not fit into any of the categories.

Personality Category

The Personality category includes MBTI and the Big Five Model [34] (see Table 1 in the Appendix for further illustration.). The Big Five Model is a taxonomy that arranges individual personalities into a hierarchical structure, categorized according to five dimensions: Extraversion, Agreeableness, Conscientiousness, Neuroticism, and Openness to Experience. The research papers ([36], [24], [32]) had different scopes. For example, Calefato et al. [36] evaluated automatic personality detectors based on the Big Five Model, also known as the Five Factor Model, in the field of software engineering for improving team composition. The aim was to determine the accuracy of the data and the feasibility of bringing personality detection to a larger scale, in addition, they replicated two studies in software engineering to examine the effect of selecting a particular tool on the validity of results. On the other hand, Gilal A. et al. [24] investigated the influence of MBTI personality type, gender and team roles on team performance. They employed four dimensions of MBTI characteristics (Introverted/Extroverted, Sensing/Intuitional, Thinking/Feeling and Judging/Perceiving) to construct a model. In addition, Gilal A. et al. [32] examined the role of programmers, focusing on introverted and extroverted traits of MBTI, and their gender. The experimental subjects for their experiments varied among the studies. For example, Calefato et al. [36] evaluated personality detection tools and replicated software engineering studies using individual developers. On the other hand, Gilal et al. [24] developed and evaluated their model using data from university students, while Gilal et al. [32], used data from both university and industry sources.

We identified a tendency to employ MBTI, over the Big Five Model. The MBTI model's popularity and accessibility may be the reason for its extensive use [61], [62]. However, recent studies have shown that it lacks fidelity and do not recommend using [63], [64], [65]. On the other hand, the experimental subjects are from academia or industry, depending on the accessibility of the researchers to these subjects.

Algorithms Category

On the other hand, the Algorithms category comprises the Decision Tree [37], the Johnson Algorithm [38], and the Genetic Algorithm [39] (see Table 2 in the Appendix for further illustration.). The decision tree is a classification technique for making decisions based on the input data. In contrast, the Johnson Algorithm, on

the other hand, is an optimization technique designed for solving complex scheduling problems. Lastly, the Genetic Algorithm is an optimization and search algorithm inspired by principles of natural selection.

Gilal A. et al. [24] developed a decision tree model to predict effective team performance based on personality traits determined by the MBTI, team roles (programmer and team leader), and gender. They used the C4.5 algorithm [66] and the 10-fold cross-validation method [67], achieving a prediction accuracy of 70.78%. Similarly, Gilal A. et al. [32] created a model for predicting the most suitable personality trait (Introvert and Extrovert pairs of MBTI) and gender for the programmer role. In this paper, they used two approaches. In the first approach, the decision tree method was used, employing the C4.5 algorithm (specifically the J48 [68] implementation), which resulted in a prediction accuracy of 58.58%. In the second approach, the Genetic and Johnson algorithms were applied, with prediction accuracies of 70.74% for both.

Both studies focused on prediction. Gilal A. et al. [24] aimed to predict effective team performance, while Gilal A. et al. [32] aimed to predict the most suitable personality trait for the programmer role. On the other hand, the two papers ([24], [32]) considered algorithms with the role and gender of the team members. In particular, the role of programmer, except for Gilal A. et al. [24], who also considered the role of team leader. Regarding the metrics used in the research papers, prediction accuracy was the primary metric calculated in all papers ([24], [32]), only Gilal A. et al. [32] identified 70% accuracy as an acceptable performance benchmark. Gilal A. et al. [24] also considered validation methods such as K-fold cross-validation for the decision tree algorithm. In contrast, Gilal A. et al. [32] employed a data division of 70% for training and 30% for testing.

The most used algorithm method was the Decision Tree in combination with other factors such as MBTI, gender and team role. Similarly, it was found that prediction accuracy was frequently used, sometimes with consideration of validation methods. Algorithms have been found to produce positive results in predicting optimal combinations of personalities and team performance.

Tools

It includes tools that employ algorithms for personality detection, such as the IBM Watson Personality Insights [45], Twitpersonality [46], LIWC (Linguistic Inquiry and Word Count) [47], and Personality Recognizer [48] (see Table 3 in the Appendix for further illustration.). The LIWC is a commercial text-analysis software designed to examine the relationship between language use and personality traits. On the other hand, the IBM Watson Personality Insights is a commercial solution that uses a machine-learning methodology whose outcome is a scale of likeliness of a personality trait of the Big Five traits. In contrast, the Personality Recognizer is a method for predicting personality traits given an analysis of written language. Lastly, Twitpersonality is another method for predicting personality traits (determined by the Big Five) based on a document written by a user.

These tools differ in their approach, techniques, and features. For instance, LIWC and Personality Recogniser use a 'top-down' [69] or closed vocabulary approach, which refers to a predetermined set of words that a system can recognise. In contrast, IBM Watson Personality Insights and Twitpersonality use a 'bottom-up' [69] or open vocabulary approach, which enables the system to recognise a wider set of words. However, the techniques used by all the tools differ. For instance, LIWC counts the frequency of specific words in a text, while IBM Watson Personality Insights employs machine learning [70]. Personality Recogniser uses regression models [71] and Twitpersonality uses support vector machines [72]. Regarding features employed, LIWC uses a closed vocabulary, whereas IBM Watson Personality Insights uses an open vocabulary and word embedding. Personality Insights uses both LIWC and the Medical Research Council database [73]. Similarly to IBM Watson Personality Insights, Twitpersonality also uses word embedding.

Based on the information provided, we observed that there is an increasing trend in utilising existing tools for conducting experiments on team composition. This shift in focus allows for the allocation of resources towards other variables such as experimental subjects or input data for the tools. Similarly, it is important to know the problem's characteristics and landscape to select the appropriate tool for our needs.

Others Category

The Others category includes Team Roles, Gender, the Umbrella Traversal Model [49], and COCOMO. The Team Roles consisted of assigning each team member a role; authors assigned roles to members who have the expertise and the technical knowledge. The roles a member could play in the project include programmer/developer, designer, and tester, among others. For instance, Gilal A. et al. [29] only explored the role of the team leader. Gender also plays an essential role in team composition. Researchers as Gilal A. [32],

have examined the suitability of different genders for programmer roles. In contrast, the Transversal Umbrella Model approach is a metamodel aimed at identifying the optimal combination of several factors [49]. Anil Kumar S. [49], explored the possibility of the Transversal Umbrella Model to construct the most suitable pair of programmers based on time to travel from one point to another. The author defines it as an optimization model that consists of efficiently moving four people with varying walking, in terms of the time needed to get from the starting point to the destination, during rainy weather. This must be accomplished with only one umbrella that can accommodate a maximum of two people simultaneously. Finally, the COCOMO was employed by Gulzar W. [22], to define the number of people in each of the development phases, as described above. In the 'other' category of methods, we identified various techniques for team building. However, we did not observe any trend in the use of these models during the implementation phase. It would be worthwhile to evaluate the methods in this category in contexts beyond those established by the research. Further to the analysis of the phases, let us look at the testing phase. In the testing phase, Gulzar W. [52] followed the same approach as in the above phases. The output of this method was the number of staff needed, determined by the effort and the duration of this phase, as mentioned earlier. Due to a lack of contributions in the testing phase, we identified an area of opportunity in investigating methods for composing teams in this phase. Concurrently, this problem could be attributed to a deficiency in interest regarding software testing [74], [75].

For papers classified as Software Development, we will employ the previously used classification. The categories are as follows: personality models, algorithms, tools, and others.

Personality Models Category

The category of personality models includes various models, such as MBTI, Five Factor Model, Belbin's Roles [41] and Keirsey Personality Type [44] (see Table 4 in the Appendix for further illustration.). Research on team composition for software development that has used these personality models has taken different approaches. For instance, in the papers using the MBTI, Gilal A. et al. [25] aimed to find the ideal personality combination for the role of the programmer with the respective team leader and combinations of programmer personalities considering their gender for software development teams. In contrast, Jaafar et al. [27] were concerned only with the selection of a software development team leader according to their personality type. Similarly, Gilal et al. [29] aimed to gain a deeper understanding of the role of the development team leader based on the personality types of team members, while considering their gender and role as programmers.

In contrast, Gilal A. et al. [28] aimed to develop a model that would identify the combination of personalities that would result in the best team performance, considering that a team should consist of one leader and four programmers. The aim was to develop a classification technique for effective and ineffective teams, considering the combination of personal factors of team members such as personality type, gender, and team role, which would contribute to the construction of the model. Alternatively, Gilal A. et al. [26] investigated whether there were differences in project success between personality preferences by applying the Role Assignment Methodology for Software Engineering Teams and using the Myers-Briggs Type Indicator (MBTI) to measure personality preferences. Meanwhile, Lewis T. et al. [76] investigated the impact of a subset of MBTI (problem-solving preferences) on the performance and level of conflict within a software development team.

In their view, Gilal A. et al [30] aimed to identify the optimal combination of personality types, based on the MBTI, for team leaders and programmers, considering their gender. In contrast, Farhangian M. et al. [31] aimed to develop a model to predict team composition and explain the team process by determining the impact of different variables, such as personality types, inter-member relationships, and previous team experience. Qamar N. et al. [35], used the Five Factor Model to assign weights to the five personality traits defined in it, creating a metric called the "Weighted Team Homogeneity Index". In contrast, Pieterse V. et al. [43] employed Keirsey's Personality Type and Belbin's Roles to create a model that considers the diversity of members' personalities. This model guides experimental subjects in the team-building process to reach peak productivity early and achieve higher team performance. Licorish et al. [42] also used Belbin's Roles, they aimed to create a tool for building software development teams that considers team members' personalities. In papers that use personality models to construct software development teams, there is a tendency to use the MBTI model in combination with factors such as team role and gender. This trend may be due to the accessibility and familiarity of the MBTI [61], [62]. However, all these methods aim to develop models or tools to assist in team creation. This tool could aid software engineers, team leaders, engineering educators, or anyone seeking improved results in team-produced software products.

In terms of personality type coverage, four ([27], [29], [76], [42]) out of the eight papers using MBTI considered only pairs of personality types, while the remaining four papers ([25], [28], [30], [26]) considered all MBTI personality types. Among the papers that focused on pairs, Lewis T. et al. [76] considered the sensing-thinking, intuition-thinking, sensing-feeling, and intuition-feeling, while the other three papers ([27], [29], [42]) focused on pairs such as introversion-extroversion, sensing-intuition, thinking-feeling, and judging-perceiving. However, the two papers ([43], [42]) that used Belbin's Roles, Pieterse V. et al. [43], who also included Keirsey's Personality Type, and Qamar N. et al. [35], who employed the Five Factor Model, considered all personality types as defined by their chosen models.

When discussing personality types, it is common to use the MBTI in two ways: either by focusing on specific pairs of personality types or by using all personality types as defined by the model, resulting in a combination of all types. In contrast, the papers reviewed for this category considered using all personality types defined by the models when building teams.

Algorithms Category

The algorithms used to form teams include genetic algorithm [39], Johnson's algorithm [38], principal component regression [50], logistic regression [52], decision tree [37], shortest path [53], rule-based model [54], naive bayes classification [55], standard voting [56] and voting with object tracking [56] (see Table 5 in the Appendix for further illustration.).

The research that used the Genetic Algorithm covered various scopes. For example, Gilal A. et al. [25] aimed to determine the optimal combination of personality types for the roles of leader and programmer. They used Johnson's Algorithm, standard voting, voting with object tracking, and Naïve Bayes classification techniques. Similarly, Gilal A. et al. [30] aimed to identify the personality patterns of programmers monitored by team leaders. Various methods were used, including the Genetic Algorithm, Standard Voting (SV), Voting with Object tracking, and Johnson's Algorithm. In contrast, Gilal A. et al. [29] developed a rule-based team composition model using two approaches. The first approach was a descriptive examination using a decision tree, while the second approach was predictive experiments using genetic and Johnson algorithms. Similarly, Gilal A. et al. [32] also used the same two approaches to create a cost-effective team for software development.

On the other hand, John B. et al. [51] aimed to create a model using principal component regression based on the Six Sigma methodology. The model would significantly reduce the time taken to solve development tickets. This would enable the team leader to strategically choose the team's size and minimum experience of the development team, considering the capacity and experience of the team members to solve the tickets. In contrast, Gilal A. et al. [28] aimed to create a model for composing software development teams based on team members' roles, personality types, and genders, using logistic regression, decision tree, and rough sets theory. Sahin Y. [40] proposed a new algorithm for team composition in software engineering courses within the academy. The algorithm considers the preferences in team composition of both teachers and students (in this model the teacher or student chooses the selection of the team) and employs the shortest path and agglomerative clustering [77] to establish efficient connections between students, a test was conducted to determine their preferences for working with other students.

Regarding the measurement of algorithm results in articles, different studies have used different metrics. Four different articles ([25], [29], [30], [32]), which used genetic algorithms, all used prediction accuracy with team performance (in terms of efficiency) to measure their results. Similarly, Gilal A. et al. [28], while employing other algorithms, also used prediction accuracy and team performance (in terms of efficiency) to measure their results. John B. et al. [51] measured their results in terms of model accuracy and process performance (measured in the process performance index [78]), while Sahin Y. [40] measured team performance (in terms of project grades, concerns regarding team members, team restlessness, time needed for team building, alignment with meeting schedules, timely project completion, and the suitability of the project concerning course objectives.).

Decision trees are increasingly being used in the algorithms used at this phase. Within the scope of the papers that employed algorithms, a strong tendency towards creating models or algorithms for building software development teams was identified. Similarly, some papers propose identifying, determining, or classifying individuals based on their gender, role, or personality type to find the optimal combination of team members. In the comparisons of the metrics used, we noticed a tendency to use more than one metric. Commonly one focuses on measuring the prediction results of either the algorithm or the model, and another on measuring the outputs of the teams that were formed based on the algorithm.

Tools Category

In the category of tools used in the papers, we identified the use of the "Role Assignment Methodology for Software Engineering Teams," a tool developed by Martínez L. et al. [79] and employed by Gilal A. et al. [26]. This tool employs sociometric [80] and psychometric techniques [81] that consider the personality [82], skills, and software roles of team members [79]. The main steps for its application are assessment of skills and abilities, application of personality assessments, conducting individual interviews, application of sociometric techniques, assignment of team responsibilities, and monitoring of the fulfilment of team roles as defined by Martínez L. et al. [79].

Others Category

A variety of methods have been considered for team building, including proposed methods such as SaiCho [57] and DELPHI [58], practical constraints, skills and their distribution among the team, personal criteria, motivational factors, the rough fuzzy model [27], roles, gender, experience, expertise, heterogeneity, expert opinion, rough set theory [28], and qualitative and quantitative methods. Understanding the available methods for building software engineering teams provides us with a broader understanding of the current knowledge in this area. Researchers can use or adapt this knowledge to suit our specific needs and contexts. This range of team-building methods during software development suggests that there are multiple ways to form teams, regardless of whether there are experimental individuals to work with.

RQ2 What are the experimental settings commonly used by development phases for team composition in the field of software engineering?

We categorized the experimental settings into two main groups: Educational and Industrial (see Figure 3 in the Appendix for further illustration.). The 'Educational' group includes settings such as courses, undergraduate experimental subjects, and universities. The 'Industrial' group refers to experiments conducted in companies, with employees or former students. Additionally, we use the "N/A" category to denote articles that did not conduct real-life experiments but rather relied on calculations or predictions. In the Requirements Phase, papers did not require experimental settings, as Gulzar W. [22] and Iqbal M. et al. [59] used methodologies based on cost estimation and job requirements analysis. However, in the Design Phase, one paper [33] employed 'Educational' settings. During the Implementation Phase, two papers each ([24], [32] and [32], [36], respectively) employed the "Educational" and "Industrial" settings, while two papers ([22], [49]), analogously in the "N/A" category did not use an experimental setting. Similarly to the requirements phase, experimental settings were not necessary as testing on people in academic or industrial settings was not required.

In the Testing Phase, one paper did not require experimental settings, as Gulzar W. [22] employed COCOMO, as previously mentioned. In contrast, for software development, the predominant experimental setting was "educational," appearing in thirteen articles ([25], [26], [27], [28], [29], [30], [35], [40], [43], [57], [76], [83], [84]). Meanwhile, "industrial" was employed in seven articles ([35], [42], [51], [58], [85], [86], [87]), while one paper [31] fell into the "N/A" category with no specified experimental settings.

There is a clear educational tendency in research aimed at improving the education of future software engineers, rather than improving established industry processes. This may be because classroom environments are more accessible than offices where it is difficult to get permission to conduct experiments [88][89].

RQ3 What are the characteristics of team members by development phases for team composition in the field of software engineering?

The team members' characteristics were classified into seven categories (C): 1. Age range, 2. Country of origin., 3. Gender (Female or Male), 4. Level of expertise (Undergraduate/Engineer), 5. Role performed, 6. Previous experience, and 7. Language. The papers ([22], [59]) classified in the Requirements Phase did not provide detailed information about the characteristics of their experimental subjects, limiting the scope of the qualitative analysis. However, in their paper, Iqbal M. et al. [59] stated that only job descriptions for requirements analysts/engineers would be considered, regardless of age, gender, or any other categorisation used in this question. Similarly, for the papers ([33], [22]) classified in the Design Phase, Hunter K. [33] provided information about the experimental subjects in his work. The participants were first-year students at Tennessee Tech University [90] in the USA who would play the role of designers in design teams. No information was provided regarding the age range, gender, previous experience, or language used for communication within the

student group. Although the university is based in the USA, this paper will not assume that the students are American or that they communicate in English. On the other hand, Gulzar W. [22], given the scope of his work, did not consider any characteristics of the team members and was limited to collecting data on the effort required to conduct the work of each phase of the software development.

During the Implementation Phase, Gilal A. et al., in both papers [29], [32], conducted their experiments with senior students at "Universiti Utara Malaysia" (UUM) [91] in Malaysia, as mentioned. In contrast, Calefato F. et al. [36] worked exclusively with engineers. Regarding gender, only Gilal A. et al. [29] and Gilal A. et al. [32] considered the gender of their participants. only Gilal A. et al. [29] obtained a sample of fourteen males and eighteen females, while Gilal A. et al. [32] worked with sixty females and thirty-seven males. On the other hand, the papers analysed distinct roles. Gilal A. et al. [29] focused on the role of the team leader, while Gilal A. et al. [32] analysed the role of the programmer. Calefato F. et al. focused on software developers and were the only ones to consider previous experience with software engineering. However, neither Anil K. [49] nor Gulzar W. [22] mentioned any characteristics of their experimental subjects. Similarly, none of the papers considered the age and language of the participants at this stage. In contrast, in the Testing Phase, Gulzar W. [22] did not concentrate on the characteristics of the individuals in the software development team. Instead, it focused on collecting objective data on the activities being performed in each phase of software development. Therefore, no information regarding the team members is available.

Regarding Software Development, all papers ([25], [28], [35], [83], [85]) that mentioned the age of their experimental subjects reported subjects older than nineteen. The average age of the experimental subjects in Nascimento N. et al. [83] was twenty-one point five years. In Gilal A. et al.[25] and Gilal A. et al. [28], the subjects were older than twenty years, while in Qamar N. et al. [35] and Marsicano G. et al. [85], the subjects ranged from twenty-four to thirty years and from nineteen to sixty-two years, respectively. In terms of country category, only Marsicano G. et al. [85] explicitly stated that their experimental subjects were Brazilian engineers. Gilal A. et al. [25], [28], [29], [30], [92] and Jaafar J. et al. [27] conducted their experiments in Malaysian universities, Qamar N. et al. [35] conducted their experiment in both academia and the software industry in Pakistan, while John B. et al. [51] conducted their project in an Indian company, and Varona D. et al. [58] had software development experts from the industry in Cuba. In contrast, Da Silva F. et al. [86], [87], who also conducted research in the industry, focused their research on the Brazilian software industry, while Pieterse V. et al. [43] conducted their experiment at the University of Pretoria in South Africa [43].

All papers for Software Development considered males and females in different proportions but with a higher proportion of male participants. For instance, Nascimento N. et al. [83] had forty-one male and six female participants, Dzvonyar D. et al. [84] had sixty-eight male and twelve female participants, Marsicano G. et al. [85] had two hundred forty males and eighty-six female participants, Gilal A. [28] had fifty males and fifty-five female participants, Sahin Y. [40] had Two hundred forty-eight males and seventy-nine female participants, Lewis T. et al. [76] had thirty-four male and four female participants, and Qamar N. et al. [35] had thirty-three male and two female professionals, as well as one hundred ninety-seven male and eighteen female students. It is worth noting that Gilal A. et al. [25], [29], [30] and Jaafar J. et al. [27] did not report the gender distribution, and Saini J. et al. [57] did not mention the gender of their participants.

The number of papers employing students was twelve ([25], [26], [27], [28], [29], [30], [40], [43], [57], [76], [83], [84]), while the number of papers employing engineers was only six ([42], [51], [58], [85], [86], [87]). However, Qamar N. et al. [35] used both student and engineer participants. The most expected roles for participants were team leaders in eight papers ([25], [26], [27], [28], [29], [30], [51], [85]), programmers in six papers ([25], [26], [28], [30], [35], [83]), and developers in six papers ([31], [42], [58], [84], [86], [87]). Two papers ([35], [83]) analysed the role of designers, while analysts and testers in the same paper [35], and one paper [51] mentioned both software engineers and quality assurance professionals.

However, previous experiences mentioned in papers include work experience (mentioned in five papers [35], [58], [83], [84], [85]), completion of programming courses (mentioned in four papers [25], [27], [28], [92]), specific university semesters (mentioned in two papers [29], [76]), studying computer or software engineering (mentioned in one paper [40]), programming languages (mentioned in one paper [31]), or models (mentioned in one paper [43]). Additionally, one paper [87] on professional maturity and management styles. The language needed for subjects was Portuguese in Marsicano G. et al. work [85][85]. Meanwhile, Dzvonyar D. et al. [84] considered subjects who speak German, English and non-German speakers.

The experimental subjects used by the researchers are usually students. Therefore, they should participate in as many roles as possible in the development of the software. In the student population, researchers got that they should be at least 20 years old, as the personality is more defined. We also found a wide age range for the professionals or engineers involved in the experiment. Moreover, there is still an imbalance in the STEM [93] population, as many men were observed in most of the experiments compared to the number of women involved.

RQ4 What are the hard and soft skills of the team members by development phases for team composition in the field of software engineering?

It is noteworthy that to maintain homogeneity among the soft skills described in the papers, we classified them according to the taxonomy of soft skills by Mahasneh J. et al. [94]. Similarly, some papers do not explicitly state the soft and hard skills required for their studies, resulting in a variation in the total number of papers in each section.

Requirements Phase. Hard Skills: Iqbal M. et al. [59] identified the hard skills required for a requirements analyst/engineer based on job descriptions for the role. The identified hard skills include: The task involves analysing the current systems in place for the client, translating their requirements into project briefs, identifying and evaluating potential solutions, creating reports on the feasibility of system/software to create an acceptable product, overseeing the development of new systems, presenting logical and innovative solutions for user systems, and offering proposals for modifying or replacing existing systems. Meanwhile, Gulzar W. [22] described the role of a systems analyst, which considers knowledge in both requirements and software design. However, for this phase, we will only consider the hard skill of requirements engineering as it is most aligned with this phase. Soft Skills: Regarding soft skills, there was no clear trend towards the importance of a specific skill. Iqbal M. et al. [59] considered teamwork and collaboration, planning and organisation, communication and social intelligence to be soft skills for the role of requirements analyst/engineer. Gulzar W. [22] considered workplace thinking to be a crucial skill for systems analysts.

Design Phase. Hard Skills: As a systems analyst, Gulzar W. [22] highlighted the importance of possessing design skills, including system design, human-computer interaction, usability engineering, software and software-hardware integration, reuse techniques, optimization, semantic-preserving transformations, and proficiency in specific programming languages and debugging techniques. Soft Skills: Hunter K. et al. [33] considered planning and organizing as the soft skills for the design phase, while Gulzar W. [22] gave a higher weight to workplace thinking.

Implementation Phase. Hard Skills: Coding was the most frequently mentioned hard skill in three research papers ([24], [32], [49]). On the other hand, Gulzar W. [22] stated that system programmers require hard skills in programming languages, data structures, database systems, operating systems, software architecture, Petri nets, complexity theory, computer graphics, linguistics, parsing theory, practical experience in a specific programming language and domain, computability theory, set theory, predicate logic, formal proof techniques, and Turing machines. Soft Skills: Planning and organising were the most frequently mentioned soft skills in three of the papers ([24], [36], [49]), followed by teamwork and collaboration, as well as communication, which were each mentioned in two papers, ([36], [49]) and ([22], [49]), respectively.

Testing Phase. Hard Skills: Gulzar W. [22] defined in their research that a system tester should possess hard skills such as previews, assessments of readiness, walkthroughs, inspections, audits of software projects, tracing of requirements, quality function deployment, techniques for software testing, tools for software testing, proofs of correctness, techniques for defining/improving processes, statistical process control, and innovations in technology. Soft Skills: Gulzar W. considered teamwork, collaboration, and communication as soft skills.

Software Development. Hard Skills: Coding was the most frequently mentioned hard skill in the papers, referenced in five of them ([24], [25], [32], [83], [84][1]. T). Followed by domain expertise and experience, which were each mentioned in three papers ([51], [57], [58]) and ([35], [51], [58]), in that order. Other skills that were mentioned less frequently included technical profile (with no details mentioned) stated in two papers ([86], [87]), task management, generating solutions, and database system concepts, which were each mentioned in one paper [29], [42], [26], respectively. Soft Skills: The soft skills most frequently mentioned in the literature are social intelligence, communication, teamwork and collaboration. These skills were identified in five papers each ([25], [29], [42], [51], [57]), ([25], [31], [40], [43], [87]), and ([31], [35], [40], [51], [87]), respectively.

Self-intelligence and workplace thinking were mentioned in three papers each ([25], [29], [42]) and ([43], [51], [86]), sequentially. Two papers ([40], [51]) mentioned workplace professionalism, as well as conflict resolution and negotiation, were mentioned in two papers ([40], [87]). Similarly, skills not so frequently mentioned are workplace diversity, planning and organising, workplace ethics and productivity, as stated in one paper each ([51], [40], [40], [86] respectively).

Successfully managing the software development process is a necessary skill for any software engineer. It is not surprising, therefore, that researchers often require experimental subjects to have the programming skills to complete the project. Technical skills are essential for task completion and project success, but they cannot exist without soft skills. “Teamwork and collaboration" and "communication" were found to be the most important soft skills in research papers. If each team member can communicate, work in a team, and collaborate effectively, the team's performance will improve.

RQ5 What are the commonly used team sizes by development phases for team composition in the field of software engineering?

In their research, different authors have defined different team sizes (see Figure 4 in the Appendix for further illustration.). For instance, during the Requirements Phase, Gulzar W. [22] determined that the requirements team should consist of three people. In contrast, during the Design Phase, Gulzar W. [22] recommended a team of five people, while Hunter K. [33] suggested a team of seven to eight people. For the Implementation Phase, Anil K. [49] proposed a two-person implementation team, while Gulzar W. calculated that the optimal team size is seven people. In contrast, for the Testing phase, Gulzar W. [22] determined that the ideal testing team should consist of four people.

Regarding Software Development, Nascimento et al. [83], John et al. [51], and Gilal et al. ([28], [29], [30]) all recommend a team of five members. Pietersen et al. and Lewis et al. [76] suggest a range of four to five members, while Gilal et al. [25] suggest a range of four to six members. However, other authors have suggested the use of larger teams. For instance, Qamar N. et al. [35] had a team of six members, while Dzvonyar D. et al. [84] had a team of seven to eight members in their development team. We found that the average team size is five members. Therefore, a small team is more efficient [95] and five members is the ideal team size [96]. Depending on the context, researchers could assign each member a different task or role, and it might be possible to work in smaller groups to reduce the workload.

6 Conclusions

This work provides updated information about team composition in the software engineering field, based on the software development phases. Through a systematic literature review, we explored the state of the art and gained significant knowledge in team composition. We found information about the methods for building teams used by researchers, characteristics of those teams, average team size, metrics to measure the performance of the method used, necessary technical and soft skills for the members, and the environmental settings where researchers conducted their experiments.

The main results of this work include the methods used to build software development teams. The following methods were used for each phase: MBTI and COCOMO for the requirements phase, COCOMO and Experimental Team-Building Program for the design phase, MBTI and decision tree for the implementation phase, COCOMO for the testing phase, and MBTI and decision tree for the software development phase. Therefore, the most frequently used method across all phases was MBTI for creating development teams. The reason for this may be that the MBTI is a more well-known personality model that is already accessible to the public [61], [62]. Despite being not recommended by many authors due to its lack of fidelity [63], [64], [65] this model is widely used in the field of team building. However, the distribution of experimental settings varied across the phases. During the requirements phase, no experimental settings were used. In the design phase, only educational settings were experimented. In the implementation phase, educational, industrial, and no settings were equally used among the papers classified in that phase. Finally, during the testing phase, no experimental settings were needed. For the Software Development phase, the experiments were mostly conducted in educational settings. This trend may be due to the accessibility of each environmental setting and the scope of the researchers' work.

In all phases, team members should possess skills in programming, teamwork, collaboration, and communication, both soft and hard. The papers had an average team size of five members. This is consistent with the literature which indicates that a smaller size is more efficient and ideal [95], [96].

Team composition in software engineering is emerging, which provides a major possibility for future investigation and the development of innovative studies. Further research can expand the body of knowledge by developing team-building methods, making use of AI algorithms, and applying the current findings to different software life cycles. This mapping study offers first contact with the building team's area. By building optimal team composition and taking advantage of current methods, software companies, and students can develop high-quality products while improving performance and keeping a competitive advantage in a constantly evolving business.

Appendix

https://alumnosuady-my.sharepoint.com/:b:/g/personal/a15001585_alumnos_uady_mx/EY3PhuhYKLtCvl0J7hVEQoUBoihgyIjPFoCUvUVtXqVSiA?e=GNrj8u

Bibliography

1. Phillips, J., & Morris, S. (2014). *Strategic Staffing* (3rd ed.). Pearson.
2. Cooke, N. J., Hilton, M. L., Science, C. on the S. of T., Board on Behavioral, C. and S. S., & National Research Council. (n.d.). Team Composition and Assembly.
3. Katzenbach, J. R., & Smith, D. K. (2015). *The Wisdom of Teams: Creating High-Performance Organizations* (p. 387).
4. Sharma, S., Sarkar, D., & Gupta, D. (2012). Agile processes and methodologies: A conceptual study.
5. Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. *International Journal of Engineering and Technology, 2*(5), 2049–3444. Recovery of http://iet-journals.org/archive/2012/may_vol_2_no_5/255895133318216.pdf
6. Costa, A., et al. (2020). Team formation in software engineering: A systematic mapping study. *IEEE Access, 8*, 145687–145712. https://doi.org/10.1109/ACCESS.2020.3011123
7. Aryanee, D., Zainal, P., Razali, R., & Mansor, Z. (2020). Team formation for agile software development: A review. *International Journal of Computer Science, 10*(2).
8. Restrepo-Tamayo, L. M., Gasca-Hurtado, G. P., & Muñoz, M. (2022). Building effective software development teams based on the personality of its members. *Communications in Computer and Information Science, 1646*, 688–703.
9. Zahidul Islam, B. A. K. M., Ferworn, A., & Zahidul Islam, A. (2020). A comparison between agile and traditional software development methodologies.
10. Jiménez-Hernández, E. M. (2012). Metodología híbrida para desarrollo de software en México. *CICIC 2012*.
11. Aguilar, R. A., Díaz, J. C., Ucán, J. P., & Quiñones, Y. O. (2019). Exploring the influence of Belbin's roles in software measurement: A controlled experiment with students. *Advances in Intelligent Systems and Computing, 865*, 69–79.
12. Aguileta-Güemez, A. A., Ucán-Pech, J. P., & Aguilar-Vera, R. A. (2017). Explorando la influencia de los roles de Belbin en la calidad del código generado por estudiantes en un curso de ingeniería de software. *Revista Educación en Ingeniería, 12*(23), 93. https://doi.org/10.26507/REI.V12N23.742
13. Alshamrani, A., & Bahattab, A. (n.d.). A comparison between three SDLC models: Waterfall model, spiral model, and incremental/iterative model. Recovery of www.IJCSI.org
14. Universidad Autónoma del Carmen, Cánepa, A. A., Christian, S., & García González, E. (n.d.). Comparativas de los modelos de ciclo de vida.
15. Mohammed, N., Munassar, A., & Govardhan, A. (2010). A comparison between five models of software engineering. *IJCSI International Journal of Computer Science Issues, 7*(5). Recovery of www.IJCSI.org
16. Kitchenham, B., & Charters, S. M. (2007). *Guidelines for performing systematic literature reviews in software engineering*. Durham.

17. Aguileta, A. A., & Gómez, O. S. (2016). Software engineering research in Mexico: A systematic mapping study. *International Journal of Software Engineering and Its Applications, 10*(12), 75–92. https://doi.org/10.14257/IJSEIA.2016.10.12.07
18. Elsevier. (n.d.). About Scopus: Abstract and citation database. Recovery of https://www.elsevier.com/solutions/scopus
19. Gestión de la información: Uso de las bases de datos Scopus y Web of Science con fines académicos. (n.d.). Recovery of http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S1316-48212016000400003
20. Rosado, N., Agüileta, A., Aguilar, R., & Ucán, J. (2023). Composición de equipos en las fases de desarrollo de la ingeniería de software: Un mapeo sistemático. *Abstraction and Application*.
21. Gómez, A., López, M. del C., Migani, S., & Otazú, A. *COCOMO: Un modelo de estimación de proyectos de software*.
22. Gulzar, W. A. (2014). Team size optimization - A management panacea: Fact or fantasy. *Life Sciences Journal, 11*(6), 469–478.
23. Myers, I. B., Myers, M. M. H., Quenk, N. L., & Hammer, A. L. (1998). *MBTI Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator* (p. 420).
24. Gilal, A. R., Omar, M., Gilal, R., Waqas, A., Afridi, S., & Jaafar, J. (2019). A decision tree model for software development teams. *International Journal of Innovative Technology and Exploring Engineering, 8*(5s).
25. Gilal, A. R., Jaafar, J., Abro, A., Waqas, A., & Basri, S. (2017). Making programmers effective for software development teams: An extended study. *Journal of Information Science and Engineering, 33*(6), 1447–1463. https://doi.org/10.6688/JISE.2017.33.6.4
26. Gilal, A. R., Jaafar, J., Basri, S., Omar, M., & Abro, A. (2016). Impact of software team composition methodology on the personality preferences of Malaysian students. *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, 454–458. https://doi.org/10.1109/ICCOINS.2016.7783258
27. Jaafar, J., Gilal, A. R., Omar, M., Basri, S., Aziz, I. A., & Hasan, M. H. (2018). A rough-fuzzy inference system for selecting team leaders for software development teams. *Advances in Intelligent Systems and Computing, 661*, 304–314.
28. Gilal, A. R., Jaafar, J., Capretz, L. F., Omar, M., Basri, S., & Aziz, I. A. (2018). Finding an effective classification technique to develop a software team composition model. *Journal of Software: Evolution and Process, 30*(1). https://doi.org/10.1002/SMR.1920
29. Gilal, A. R., Jaafar, J., Basri, S., Omar, M., & Waqas, A. (2016). A rule-based model for software development team composition: Team leader role with personality types and gender classification. *Information and Software Technology, 74*, 105–113. https://doi.org/10.1016/j.infsof.2016.01.002
30. Gilal, A. R., Jaafar, J., Basri, S., Omar, M., & Tunio, M. Z. (2016). Making programmers suitable for team-leader roles: Software team composition based on personality types. *2015 International Symposium on Mathematical Sciences and Computing Research (iSMSC)*, 78–82. https://doi.org/10.1109/ISMSC.2015.7594031
31. Farhangian, M., Purvis, M. K., Purvis, M., & Savarimuthu, B. T. R. (2016). Modeling team formation in self-assembling software development teams. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1319–1320.
32. Gilal, A. R., Jaafar, J., Omar, M., Basri, S., & Aziz, I. D. A. (2019). A set of rules for constructing gender-based personality types' composition for software programmers. *Lecture Notes in Electrical Engineering, 520*, 363–374. https://doi.org/10.1007/978-981-13-1799-6_38
33. Hunter, K. W., Matson, J. O., & Dunn, L. R. (2002). Impact of a fifty-minute experiential team-building program on design team performance. *ASEE Annual Conference Proceedings*, 9163–9171. https://doi.org/10.18260/1-2--10401
34. McCrae, R. R., & John, O. P. (1992). An introduction to the five-factor model and its applications. *Journal of Personality, 60*(2), 175–215.
35. Qamar, N., & Malik, A. A. (2021). Determining the relative importance of personality traits in influencing software quality and team productivity. *Computing and Informatics, 39*(5), 994–1021. https://doi.org/10.31577/CAI_2020_5_994
36. Calefato, F., & Lanubile, F. (2022). Using personality detection tools for software engineering research: How far can we go? *ACM Transactions on Software Engineering and Methodology, 31*(3). https://doi.org/10.1145/3491039

37. IBM. (n.d.). ¿Qué es un árbol de decisión? Recovery of https://www.ibm.com/es-es/topics/decision-trees
38. Demaine, E., Ku, J., & Solomon, J. (n.d.). *Introduction to Algorithms: 6.006, Lecture 14: Johnson's Algorithm*.
39. Mirjalili, S. (2019). Genetic algorithm. *Studies in Computational Intelligence, 780*, 43–55. https://doi.org/10.1007/978-3-319-93025-1_4
40. Sahin, Y. G. (2011). A team-building model for software engineering courses term projects. *Computers & Education, 56*(3), 916–922. https://doi.org/10.1016/J.COMPEDU.2010.11.006
41. Belbin, R. M. (2010). *Team Roles at Work* (p. 153).
42. Licorish, S., Philpott, A., & MacDonell, S. G. (2009). Supporting agile team composition: A prototype tool for identifying personality (in)compatibilities. *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering (CHASE 2009)*, 66–73. https://doi.org/10.1109/CHASE.2009.5071413
43. Pieterse, V., Kourie, D. G., & Sonnekus, I. P. (2006). Software engineering team diversity and performance. *ACM International Conference Proceeding Series, 204*, 180–186. https://doi.org/10.1145/1216262.1216282
44. Keirsey. (n.d.). Recuperado el 24 de febrero de 2024, de https://keirsey.com/temperament-overview/
45. IBM. (n.d.). *IBM Watson Natural Language Understanding*. Recuperado el 25 de julio de 2023, de https://www.ibm.com/products/natural-language-understanding
46. Carducci, G., Rizzo, G., Monti, D., Palumbo, E., & Morisio, M. (2018). TwitPersonality: Computing personality traits from tweets using word embeddings and supervised learning. *Information, 9*(5), 127. https://doi.org/10.3390/INFO9050127
47. Pennebaker, J. W., Boyd, R. L., Jordan, K., & Blackburn, K. (2015). The development and psychometric properties of LIWC2015.
48. Mairesse, F., Walker, M. A., Mehl, M. R., & Moore, R. K. (2007). Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of Artificial Intelligence Research, 30*, 457–500. https://doi.org/10.1613/JAIR.2349
49. Kumar, S. A. (2020). Importance of proper pairing in agile teams reinforced with umbrella traversal problem. *Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2020)*, 1277–1282.
50. Deduy, I. (2019). *Regresión sobre componentes principales*. Universidad de Sevilla.
51. John, B., & Kadadevaramath, R. S. (2020). Improving the resolution time performance of an application support process using Six Sigma methodology. *International Journal of Lean Six Sigma, 11*(4), 663–686. https://doi.org/10.1108/IJLSS-10-2018-0108
52. IBM. (n.d.). *What is Logistic Regression?* Recuperado el 6 de julio de 2023, de https://www.ibm.com/topics/logistic-regression
53. Black, P. (n.d.). *Shortest Path*. *Dictionary of Algorithms and Data Structures*. Recuperado el 25 de julio de 2023, de https://xlinux.nist.gov/dads/HTML/shortestpath.html
54. Takagi, T., & Sugeno, M. (n.d.). Fuzzy Identification of Systems and Its Applications to Modeling and Control.
55. Leung, K. M. (2007). *Naive Bayesian Classifier*.
56. Olson, D., & Delen, D. (2008). *Advanced Data Mining Techniques*.
57. Saini, J. R., & Cnomal, V. S. (2017). SaiCho: A parameters-based model for team building for academic software projects. *Proceedings of the 2017 2nd IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT 2017)*.
58. Varona, D., & Capretz, L. F. (2021). Using the Delphi method for model for role assignment in the software industry. *IECON Proceedings [Industrial Electronics Conference], 2021-October*. https://doi.org/10.1109/IECON48115.2021.9589957
59. Iqbal, M. A., Shah, A., & Khan, T. K. U. (2019). Predicting most productive requirements elicitation teams using MBTI personality traits model. *International Journal of Engineering and Advanced Technology, 9*(1), 3809–3814. https://doi.org/10.35940/IJEAT.A9833.109119
60. Tuckman, B. W. (1965). Developmental sequence in small groups. *Psychological Bulletin, 63*(6), 384–399. https://doi.org/10.1037/h0022100
61. He, J. (2024). An analysis of the reasons for the popularity of MBTI personality tests. *Advances in Social Behavior Research, 5*(1), 1–4. https://doi.org/10.54254/2753-7102/5/2024036

62. Li, S. (2023). A study on the popularity of MBTI in social media platform brings self-labeling and group stereotype to college students. *Communications in Humanities Research, 10*(1), 318–324. https://doi.org/10.54254/2753-7064/10/20231357
63. Boyle, G. J. (1995). Myers-Briggs Type Indicator (MBTI): Some psychometric limitations. *Australian Psychologist, 30*(1), 71–74. https://doi.org/10.1111/j.1742-9544.1995.tb01750.x
64. Stein, R., & Swan, A. B. (2019). Evaluating the validity of Myers-Briggs Type Indicator theory: A teaching tool and window into intuitive psychology. *Social and Personality Psychology Compass, 13*(11), e12434. https://doi.org/10.1111/spc3.12434
65. Zárate-Torres, R., & Correa, J. C. (2023). How good is the Myers-Briggs Type Indicator for predicting leadership-related behaviors? *Frontiers in Psychology, 14*, 940961. https://doi.org/10.3389/fpsyg.2023.940961
66. Saha, S. (2023). What is the C4.5 algorithm and how does it work? *Towards Data Science*. Recovery of https://towardsdatascience.com/what-is-the-c4-5-algorithm-and-how-does-it-work-2b971a9e7db0
67. Brownlee, J. (2023). A gentle introduction to k-fold cross-validation. *Machine Learning Mastery*. Recovery of https://machinelearningmastery.com/k-fold-cross-validation/
68. Khanna, N. (2023). J48 classification [C4.5 algorithm] in a nutshell. *Medium*. Recovery of https://medium.com/@nilimakhanna1/j48-classification-c4-5-algorithm-in-a-nutshell-24c50d20658e
69. Eichstaedt, J. C., et al. (2021). Closed- and open-vocabulary approaches to text analysis: A review, quantitative comparison, and recommendations. *Psychological Methods, 26*(4), 398–427. https://doi.org/10.1037/met0000349
70. Wiederhold, G., & McCarthy, J. (1992). Arthur Samuel: Pioneer in machine learning. *IBM Journal of Research and Development, 36*(3), 329–331.
71. Silhavy, R., Silhavy, P., & Prokopova, Z. (2017). Analysis and selection of a regression model for the use case points method using a stepwise approach. *Journal of Systems and Software, 125*, 1–14. https://doi.org/10.1016/j.jss.2016.11.029
72. Vapnik, V. N. (1998). *Statistical learning theory*. Wiley.
73. Coltheart, M. (1981). The MRC Psycholinguistic Database. *Quarterly Journal of Experimental Psychology, 33*(4), 497–505.
74. Capretz, L. F., Harous, S., & Nassif, A. B. (2023). What UAE software students think about software testing: A replicated study. *Computers Communications and Control*. https://doi.org/10.1007/978-3-031-16684-6_7
75. de S. Santos, R., Capretz, L. F., Magalhães, C., & Souza, R. (2023). Myths and facts about a career in software testing: A comparison between students' beliefs and professionals' experience. *IEEE Software, 40*(5), 76–84. https://doi.org/10.1109/MS.2023.1018123
76. Lewis, T. L., & Smith, W. J. (2008). Building software engineering teams that work: The impact of dominance on group conflict and performance outcomes. *Proceedings - Frontiers in Education Conference (FIE)*. https://doi.org/10.1109/FIE.2008.4720498
77. ScienceDirect. (n.d.). Agglomerative clustering - An overview. Recovery of https://www.sciencedirect.com/topics/computer-science/agglomerative-clustering
78. Steiner, S., Abraham, B., & Mackay, J. (n.d.). Understanding process capability indices.
79. Martínez, L. G., Castro, J. R., Licea, G., Rodríguez-Díaz, A., & Álvarez, C. (2010). Towards a fuzzy model for RAMSET: Role assignment methodology for software engineering teams. *Studies in Computational Intelligence*, 23–41.
80. Cillessen, A. H. N., Rubin, K. H., & Bukowski, W. M. (2009). *Handbook of peer interactions, relationships, and groups*. The Guilford Press.
81. Borsboom, D., & Molenaar, D. (2015). Psychometrics. *International Encyclopedia of the Social & Behavioral Sciences: Second Edition*, 418–422. https://doi.org/10.1016/B978-0-08-097086-8.43079-5
82. ScienceDirect. (n.d.). Psychometrics - an overview. Recuperado el 24 de febrero de 2024, de https://www.sciencedirect.com/topics/social-sciences/psychometrics
83. Nascimento, N., Sales, A., Santos, A. R., & Chanin, R. (2019). An investigation of influencing factors when teaching on active learning environments. *ACM International Conference Proceeding Series*, 517–522. https://doi.org/10.1145/3350768.3353819
84. Dzvonyar, D., Alperowitz, L., Henze, D., & Bruegge, B. (2018). Team composition in software engineering project courses. *Proceedings - International Conference on Software Engineering*, 16–23. https://doi.org/10.1145/3194779.3194782

85. Marsicano, G., da Silva, F. Q. B., Seaman, C. B., & Adaid-Castro, B. G. (2020). The Teamwork Process Antecedents (TPA) questionnaire: Developing and validating a comprehensive measure for assessing antecedents of teamwork process quality. *Empirical Software Engineering, 25*(5), 3928–3976. https://doi.org/10.1007/S10664-020-09860-5
86. Da Silva, F. Q. B., França, A. C. C., Gouveia, T. B., Monteiro, C. V. F., Cardozo, E. S. F., & Suassuna, M. (2011). An empirical study on the use of team building criteria in software projects. *International Symposium on Empirical Software Engineering and Measurement*, 58–67. https://doi.org/10.1109/ESEM.2011.14
87. Da Silva, F. Q. B., et al. (2013). Team building criteria in software projects: A mix-method replicated study. *Information and Software Technology, 55*(7), 1316–1340. https://doi.org/10.1016/J.INFSOF.2012.11.006
88. Pfleeger, S. L. (1995). Experimental design and analysis in software engineering. *Annals of Software Engineering, 1*(1), 219–253. https://doi.org/10.1007/BF02249052
89. Sjøberg, D. I. K., et al. (2005). A survey of controlled experiments in software engineering. *IEEE Transactions on Software Engineering, 31*(9), 733–753. https://doi.org/10.1109/TSE.2005.97
90. Tennessee Tech University. (n.d.). About Tennessee Tech University. Recuperado el 1 de octubre de 2023, de https://www.tntech.edu/about/index.php
91. Universiti Utara Malaysia. (n.d.). History. Recuperado el 3 de marzo de 2024, de https://www.uum.edu.my/about/overview/history
92. Singh, B., & Gautam, S. (2016). The impact of software development process on software quality: A review. *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, 666–672. https://doi.org/10.1109/CICN.2016.137
93. U.S. Department of Education. (n.d.). Science, technology, engineering, and math, including computer science. Recuperado el 16 de julio de 2023, de https://www.ed.gov/stem
94. Mahasneh, J., Thabet, W., & Mahasneh, J. K. (2016). Developing a normative soft skills taxonomy for construction education. *Journal of Civil Engineering and Architecture Research, 3*(5), 1468–1486.
95. Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., & Madachy, R. (1998). Using the WinWin Spiral Model: A case study. *Computer [Long Beach Calif], 31*(7), 33–44.
96. Koch, P. (n.d.). The ideal web team [part 1]. *Digital Web Magazine*.