Received 17 May 2022, accepted 20 June 2022, date of publication 27 June 2022, date of current version 5 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3186347

RESEARCH ARTICLE

A Genetic Algorithm-Based Approach to Support Forming Multiple Scrum Project Teams

ALEXANDRE COSTA[1,2], FELIPE RAMOS[1,2], MIRKO PERKUSICH[2], ADEMAR DE SOUSA NETO[2,3], LUIZ SILVA[2,3], FELIPE CUNHA[2,3], THIAGO RIQUE[2,3], HYGGO ALMEIDA[2,3], AND ANGELO PERKUSICH[2,3], (Member, IEEE)

[1]Federal Institute of Paraíba, Santa Luzia 58.600-000, Brazil
[2]Virtus, Campina Grande 58.429-140 Brazil
[3]Electrical Engineering and Informatics Center, Federal University of Campina Grande, Campina Grande 58.429-900, Brazil

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

This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brazil (CAPES)—Finance Code 001.

ABSTRACT Forming effective teams is an essential but challenging task, especially for organizations that carry out multiple projects simultaneously, a problem known as the Multiple Team Formation (MTF) problem. The literature presents several solutions for the MTF problem, mostly modeling it as a search problem. However, the existing solutions are not suitable for Scrum projects. We addressed this gap by developing an approach composed of two main steps. First, we designed a Structured Task Model to support creating developers' profiles given their performance on past Scrum projects. Then, given a set of target projects' technology requirements and the available developers' profiles, we developed a Genetic Algorithm to form the teams for a set of target projects. We evaluated the proposed approach by comparing the teams formed by our approach with the ones formed by project managers from one organization. Our approach achieved 85% of precision when compared with the teams provided by the project managers who worked on the same target projects. We also recorded an acceptance rate of up to 75%. The significant value of precision achieved suggests that our approach can provide teams close to the project managers' expectations. In addition, our Structured Task Model offers a promising way to build technical profiles semi-automatically for Scrum developers. In future work, we intend to investigate how to complement the developers' profiles by using other types of attributes and knowledge sources.

INDEX TERMS Intelligent software engineering, search-based software engineering, genetic algorithm, multiple team formation problem.

I. INTRODUCTION

In the dynamic and competitive modern economic environment, organizations are compelled to manage multiple projects to generate revenue and maintain competitive advantage [1]. In this context of carrying out multiple projects, a challenge is to form teams, a problem known as the Multiple Team Formation problem [2], which is NP-hard. Further, since hiring and training are expensive [3], it is logical for organizations to put significant efforts to ensure that the return on investment is maximized by ensuring the best use of the resources [4]. This problem has at least two perspectives: forming a team for a new problem or changing teams during project execution.

In this context, the decision-maker must analyze the scope of each project, and the competences of the available human resources for forming teams. For new projects, the decision-maker analyzes the project's requirements and the available resources, which might be previously allocated to other projects. For existing projects, it is a continuous process, since requirements and business demands change constantly.

It might also be necessary to change teams during project execution due to turnover or to reduce costs or risks. For

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

instance, if a senior C++ developer is working in a project for which the complex C++ features are already done, it might be financially worth it for the company to swap the senior C++ developer for a junior one, allocating the senior developer to a higher risk project. Conversely, consider that a project, for some reason, gets into a high-risk state due to issues with features related to augmented reality. In this case, it might be necessary to allocate an augmented reality expert for this project.

For both cases, the decision-maker relies on his expertise, collaborates with his colleagues, and explores organizational process assets (e.g., project's lessons learned and reports). Notice that, during this process, the decision-maker may conclude that it is not possible to form the team with the necessary competences and request hiring or training. As previously discussed, the complexity related to solving this problem increases exponentially, given the number of projects and candidate members.

For organizations with a high rate of similarity between projects, there is the potential for reusing knowledge from past projects to assist in decision making. For instance, Dantas *et al.* [5] analyzed the similarity between the features of a target project and historical data to estimate effort, and Ramos *et al.* [6] have done the same to infer non-functional requirements. A similar approach is possible to address the MTF problem if the company has detailed information about the work done by each collaborator. For instance, consider project A, which consists of ten requirements, which are similar to the requirements of projects B and C. By analyzing the work done in projects B and C, it is possible to infer the technologies and types of features to be implemented for project A.

A recent systematic mapping study on team formation in software engineering [7] pointed out that several studies address the MTF problem as a search problem, modeling the competences of the human resources and the projects' requirements, and solving it using several techniques such as Genetic Algorithms (GAs), Simulated Annealing, and Dynamic Programming, and others. The existing studies rely on several types of knowledge sources to model the competences of human resources such as curriculum, social validation, and historical data. The ones that focus on historical data rely on information related to traditional project management tasks, which are estimated to last between 20 and 80 hours.

Over the past years, organizations started to apply Agile Software Development (ASD) [8]–[11], which have become mainstream [12] and have Scrum as its most popular method [13]. For Scrum, the tasks are estimated to last one day or less. Therefore, having a higher granularity related to the registered work done by the developer, when comparing to the traditional task. For instance, while a traditional task might relate the developer to a programming language or library, the task might relate him to the library's function used. Unfortunately, ASD relies mostly on tacit knowledge [14], so, usually, the information registered for a Scrum task is not enough to infer useful knowledge. This observation raises the following research question: *how to form teams, more assertively, in the context of organizations that carry out multiple projects, with a high rate of similarity between them, using Scrum?*

To address this question, we followed a two-steps approach. First, we defined a procedure to support Scrum projects by improving the usefulness of the persisted technical tasks on the project database. Such a procedure is based on adding tags to the tasks, registering their scope, and related technologies. This information is used to build a detailed profile of the developers. Plus, with the support of a US-reuse framework, such as the ones presented in Dantas *et al.* [5], Ramos *et al.* [6], and Elamin and Osman [15], it enables to predict the tasks to be developed in projects before their execution, enabling to build detailed project profiles. Second, the detailed developers' and projects' profiles are used as input into a Genetic Algorithm, which searches for the best global team formation for Scrum projects. Thus, supporting decision-makers to infer the scope of Scrum projects and form teams more assertively.

This paper details our proposed algorithm and structured model to describe Scrum tasks. Further, it reports the results of validating the algorithm with a dataset of 1063 tasks collected from 12 projects of one organization. Finally, it presents the result of evaluating the algorithm with four project managers (PMs) of the same organization.

The remainder of the paper is organized as follows. Section II presents the related work on team formation. Section III describes the approach to form multiple teams for Scrum projects. Section IV details the process to validate the proposed solution. Section V presents the process to evaluate the proposed solution. Section VI shows the threats to validity of the research. Section VII presents the conclusion and limitations of the research and future work.

II. RELATED WORK

The multiple team formation problem is a well-know problem in several and distinct domains such as soccer [16], online games [17], robotic competitions [18], [19], on-demand taxi-calling platforms [20], and others [21]–[25].

As mention before, the MTF problem is NP-hard and is commonly modeled as a search and optimization problem. Silva and Costa [26] proposed a decision model based on dynamic programming to form teams that can minimize the duration of software development projects. Gharote *et al.* [27] used a Scatter search algorithm to allocate trainees to a software project to minimize retraining and relocation costs. Crawford *et al.* [28] presented a Linear Programming approximation algorithm to select a minimal-cost team to complete a given set of tasks.

In the software domain, there is a particular field that deals with metaheuristic search-based optimization techniques to provide automated and semi-automated solutions, known as Search-Based Software Engineering (SBSE) [29]. SBSE techniques include Hill Climbing, Simulated Annealing, Genetic Algorithm, and others. However, GA stands out as the most popular choice. For instance, Costa *et al.* [30] proposed an automated approach to form multiple teams in Scrum projects using a single-objective GA. stylianou and Andreou [31] presented a multi-objective GA to optimize the project's schedule and assigning the most experienced developers to tasks.

Arunachalam *et al.* [32] used GA to form teams with based average team's cost and productivity. Other studies using GA to form teams can be found in [33], [34].

Besides the computational algorithm, studies in the literature use different types of attributes to allocate the team members such as personality traits [35]–[37] and social interactions [38]–[40]. For instance, Gilal *et al.* [36] developed a rule-based model for software development team formation based on gender and personality traits. Latorre and Suárez [40] presented a framework to allocate people to projects from a socio-technical perspective,i.e., considering the abilities a person uses to interact with other people.

Furthermore, the related work varies according to the granularity information used to perform the allocation. Ye *et al.* [41] proposed a team formation approach to recommend teammates for crowdsourcing developers. The allocation is performed based on three factors: closeness with teammates, expertise difference with teammates, expertise gain through collaboration. Zhang *et al.* [42] proposed a framework to form teams by maximizing members' skills gain and minimizing the communication costs among teammates. The allocation is based on requirements such as Security, Multimedia, Management, Hardware, Networking, and others. Costa *et al.* [30] proposed an approach to allocate multiple developers into multiple teams using data from tag-based profiles. However, the tags are applied to the projects, not to the tasks. Tseng *et al.* [43] proposed an approach that uses Fuzzy sets theory and Grey decision theory to form multi-functional teams. The allocation depends on competences such as network technique, software design, database design, quality control, hardware testing, and others. Strnad and Guid [33] presents a decision support system for project team formation. The team are formed using historical information such as experience time in programming languages, the number of projects in which the developer has previously worked, priority and status of the projects, and others.

We believe our approach differs and complements state of the art because we provide a systematic way to record structured data that allows building more reliable profiles to represent the developers' technical competences. By doing this, we can know the work done by the developers in specif level.

III. PROPOSED SOLUTION

The proposed solution consists of an approach to support the MTF process for Scrum projects. It is based on the allocation of the organization's available developers into multiple target software projects, aiming at obtaining the best global team formation. Our approach intends to assist PMs in forming project teams by providing them with suggested teams in which their members possess competences that match the projects' requirements at the maximum possible level.

Our solution is divided into two parts: (i) constructing the developers' profiles and (ii) using the profiles and projects' requirements to form the teams. To construct the developers' profiles with the level of detail aimed by our solution, we propose the structured task model, discussed in Section III-A. To build this model, we followed the taxonomy for User Stories (USs) proposed by Dilorenzo *et al.* [44]. The taxonomy enables the reuse of USs and their related assets. It adds link semantics between USs, allowing the identification of similar USs.

To form the teams, having as inputs the projects' requirements and the developers' profiles, we developed a Genetic Algorithm, discussed in Section III-B. It outputs a global team formation, i.e., optimal teams for each target project, based on their requirements.

Despite the potential to be applied to other contexts, we focused on projects that use Scrum, which was chosen due to its popularity [13]. We consider that our solution can be especially useful to a scenario with dozens or hundreds of developers distributed in dozens of projects with a level of similarity to enable the reuse of data from past projects to predict the scope of future ones.

A. STRUCTURED TASK MODEL

An essential requirement to form Scrum teams is to know the competences of the available developers, i.e., the developers' profiles. There are several knowledge sources that can be used to infer the developer's profile, such as project database, training resources, declaration, and assessment [45], [46]. We assume that the most trustworthy source is the project database. Further, we assume that to be more assertive regarding the work to do and the developers' profiles, it is necessary to have high granularity regarding the work performed by the developers. For instance, it is not enough to know if a developer has worked with Java tasks, because it might have been for Web, Desktop, or Mobile development.

Further, we assume that it is not enough to know if a developer worked with Java for Android development. We believe that there is a need for more details regarding the performed work. A valid example would be to know that a developer worked with Java for Android using the Tesseract Engine[1] for Optical Character Recognition (OCR). With such a level of detail, we could recommend this developer for a project that focuses on developing an Android application having OCR as a feature.

In Scrum, the smallest unit of work is the technical task. Other than the source code and commits, they are the project assets with more details regarding the work done by each developer. Traditionally, they are written in natural language, resulting in unstructured data and represented by small sentences. The major drawback, from a knowledge-reuse driven perspective, is that only the traditional description of tasks

[1]https://github.com/tesseract-ocr/tesseract

is not enough to infer much about the work executed (e.g., technologies used). Consequently, the competences acquired during the implementation remain tacit, hindering the potential of having a project database with enough information to allow us to determine the developers' competences at the desired level.

To address this problem, we designed the Structured Task Model (STM). It allows the developers' to systematically add useful data about the technologies used to implement tasks. Such data is then used to build the technical profiles with the required level of granularity. In this study, we opted to focus on structuring the tasks and defining the level of information necessary to allow us to form teams with the required level of assertion, deferring exploring how to automatically extract such information from commit messages and source code to future work. Further, it is worthy of mentioning that such an approach does not compromise agile values and principles. Applying such a technique must not be enforced to the team, but negotiated with them. The managers must clarify to the team that despite the effort on their part, compromising the minimum of the project's productivity, the organization becomes potentially more productive.

Figure 1 shows the format of the structured task. It is composed of two parts: (i) a standard task description, representing the work to be done, and (ii) a set of technology tags to represent the associated technologies. The processes to generate that structured tasks are detailed as follows.

1) THE TASK LABELING PROCESS (TLP)

This process consists of assigning tags to tasks to contextualize them regarding the technologies used during their development. Scrum does not provide mechanisms to contextualize these artifacts; thus, we use tags, which are element descriptors that assist in classifying content. We designed the TLP to have it applied when team members break down the product backlog items into tasks (i.e., during the sprint planning meeting and the sprint's execution). The tasks can be labeled with technology tags while being created or closed. Conceptually, technology includes tools, methods, and techniques to accomplish the tasks. Figure 1 shows the technology descriptors for contextualizing the tasks. *Layer* indicates the architectural layer related to the task (e.g., frontend or backend); *Programming language* represents the programming languages related to the task such as Java, C++, Python, Ruby, and others; *Framework* may include Express, Django, Rails, Spring, Angular, React, and others; *API* comprises Google Maps, Facebook, Stripe, Twilio, Slack, and others. *Persistence* can be MySQL, Oracle, SQLServer, PostgreSQL, MongoDB, and others. Additionally, it is possible to add as many tags as necessary to contextualize the task, such as markup (e.g., HTML and XML) and style sheet languages (e.g., CSS).

2) TASK CATEGORIZATION PROCESS (TCP)

Different projects usually have hundreds of tasks with the same purpose, but often, these tasks are documented with different descriptions. For instance, a developer who participated in several projects might have tasks such as *Develop authentication page*, *Generate view for authentication*, and *Build authentication page* in his development history. These tasks have the same goal, which is to create an interface allowing the user to log in to the system. However, computational algorithms have difficulty in recognizing different text sentences as the same [47], [48]. Natural Language Processing approaches [49]–[51] have been proposed over the years. However, they depend on the quality and completeness of the information presented in the text. Therefore, the TCP aims to categorize tasks according to their purpose. The defined category adds link semantics to the tasks, enabling them to be compared, and consequently, reused [15]. We further discuss the process of reusing them in Section III-B. The application of this process is analogous to the TLP, but instead of applying tags to the task, the developer team must choose the most appropriate standard task description to represent its purpose.

The team must apply the STM during the project's development cycles; thus, the proposed solution generates the profiles over time. Project profiles are composed of sets of structured tasks. Similarly, developer profiles consist of all the structured tasks the developer implemented, considering all the projects in which he was a member.

In this study, we model the developers' competences as knowledge and skills. Knowledge is information obtained through sensory input (i.e., it refers to learning concepts, principles, and information related to a particular subject). We represent the developer's knowledge using technology tags. On the other hand, skills are the proficiencies acquired through training and practice, i.e., they refer to the ability to apply knowledge to specific contexts. We represent the developer's skills using standard task descriptions. We believe that tags can register what the developer knows (knowledge). Whenever he implements a task, we assume that, at some level, he knows the associated technologies. Although, by using only tags, it is not possible to discover on what this knowledge has been applied. Thus, we use the standard task descriptions (skills).

B. MULTIPLE TEAM FORMATION

To initiate the MTF process, the organization must define the target sprints from target projects. For a new project, it could be all the sprints. However, if the project is already under execution and management sees the need for changes due to turnover, risk, or cost management, it could be a set of the remaining sprints.

As discussed in Section III-A, we aimed to allocate members to projects considering detailed competences about the types of tasks to be executed and technologies to be applied, maximizing the coverage of the projects' needs. Hence, our approach builds the developers' and projects' profiles based on the STM (see Section III-A). It is worthy of mentioning that such an approach does not hinder the Just In Time (JIT) requirements nature of agile projects. We assumed the context

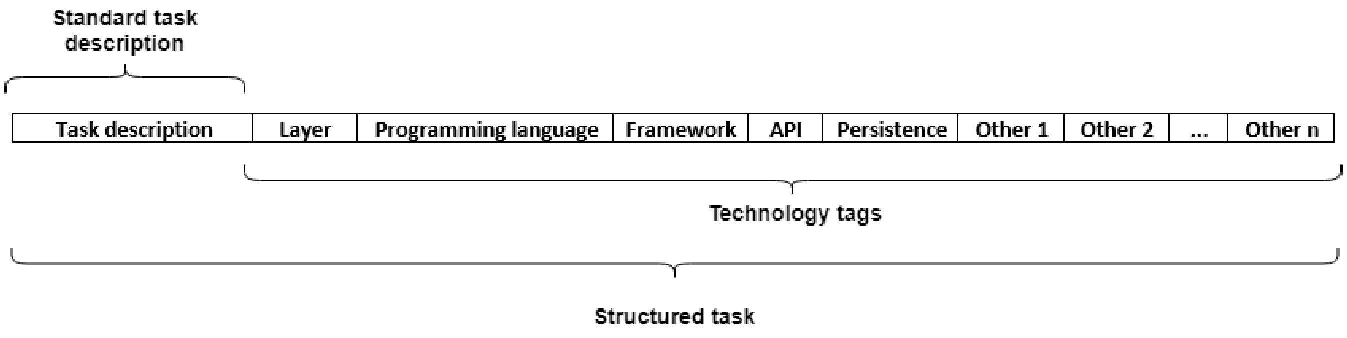


FIGURE 1. Format of the structured tasks.

of an organization that has enough similar projects enabling the reuse of requirements and related assets.

In agile projects, including Scrum, the most popular notation to document business requirements is the User Story [52]. Using an approach such as the ones presented in Dantas *et al.* [5], Ramos *et al.* [6], and Elamin and Osman [15] to add link semantics between US enable inferring the similarity between them; thus, facilitating the reuse of US for new projects. Since most agile tools automatically construct traceability between US and tasks (i.e., a one-to-many relationship), it is possible to reuse tasks having the US as the index.

Let US_A be a set of User Stories considered for project A $US_A = \{u_1, u_2, \ldots, u_n\}$, which were reused from past projects. Given that each u_i is associated with a set of tasks T_i following the STM, we could use T_i to define the needs for project A before its execution. The same reasoning can be transferred to the case in which the project is already under execution, but needs to change the project team.

Further, to allocate developers to each project, it is necessary to define their knowledge and skills. For this purpose, our approach relies on data from past projects using the STM (Section III-A). Afterward, we represent the compatibility of each developer to each project using a competence matrix (Section III-B3). We use a GA to provide global team formation (Section III-B4). We detail each of these steps in the subsections presented in what follows.

1) DEVELOPER KNOWLEDGE QUANTIFICATION

The developer's profile is composed of tags that represent his knowledge about technologies he worked in the past. To quantify this knowledge, we created Feature Vectors (FVs) based on specific sprints of the target projects. FVs are k size vectors formed by numerical values that represent the characteristics of target objects. This data structure is a popular representation in the Machine Learning area [53]. Figure 2 shows the standard structure of the FVs used in this study. Each tag key is associated to a technology tag (e.g., Tesseract Engine, Java, and Android). The FVs have values in the range [0, 1] at each index.

As previously discussed, our approach forms teams based on specific sprints of each target project. Let A be a set of

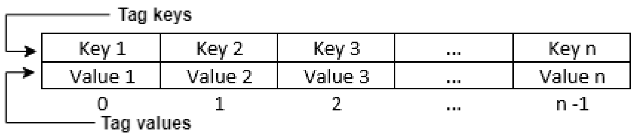


FIGURE 2. Structure of a feature vector.

target projects $A = \{a_1, a_2, \ldots, a_m\}$, where a_i represents the set of sprints considered for each target project, and D be the set of available developers $D = \{d_1, d_2, \ldots, d_k\}$, where d_i represents the profile for the *ith* developer. Given the reuse of tasks discussed previously, we can infer the tasks X_{a_i} for each $a_i \in A$.

Let $X_{a_i} = \{x_{a_{i_1}}, x_{a_{i_2}}, \ldots, {a_{i_n}}\}$ be the set of tasks for a project a_i. For each task x_{a_i}, we have a task FV ($FV_{x_{a_i}}$) and a set $FV_{x_{a_{i_{d_j}}}}$ containing an FV for each developer $d_j \in D$ representing his knowledge for the task x_{a_i}, which is defined given his profile (discussed later). $FV_{x_{a_{i_{d_j}}}}$ allows calculating the similarity calculation between a developer d_j and the task x_{a_i} in terms of the technology tags. For $FV_{x_{a_i}}$, we have a tag key for each technology associated with the project and, by default, we set the tag values to one (1).

To define each $FV_{x_{a_{i_{d_j}}}}$ for every $a_i \in A$, we analyze the profile for each $d_j \in D$. Therefore, we define $|X| * |D|$ FVs mapping each developer d_j to each target task x_{a_i} for each project $a_i \in A$. Each $FV_{x_{a_{i_{d_j}}}}$ contains the same number of tag keys as $FV_{x_{a_i}}$, because our goal is to calculate the competence of each developer with regards to the technologies needed for task x_{a_i}. To define the tag values for $FV_{x_{a_{i_{d_j}}}}$, we count how many tasks has the developer d_j implemented that were associated for each tag key contained in $FV_{x_{a_i}}$. For instance, assume that the task x_{a_i} includes the technologies Android, Java, Tesseract, and XML and the developer d_j had implemented, respectively, 30, 45, 5, and 3 tasks for each of the listed technologies. Therefore, $FV_{x_{a_{i_{d_j}}}}$ could be represented by the tuples $FV_{x_{a_{i_{d_j}}}} = \{(\text{``}Android\text{''}, 30), (\text{``}Java\text{''}, 45), (\text{``}Tesseract\text{''}, 5), (\text{``}XML\text{''}, 30)\}$.

Notice that, initially, the filled values are not within the desired range of [0, 1]. We normalize them given Equation 1, where x_{ij} is the number of tasks of the *ith* technology (i.e., key) implemented by developer d_j, max(i) is the maximum number of tasks associated with the *ith* key implemented by a developer considering all $d_j \in D$, and z_{ij} is the normalized value for the *ith* key and *jth* developer. For instance, consider that for the key "Android", the maximum value is of 150 related to developer d_a. In other words, considering that the key "Android" as the *rth* key, $x_{r_a} = 150$. Thus, by applying Equation 1, $z_{r_a} = 1$. For a developer d_b who have developer 30 "Android" tasks, we have $z_{r_b} = 0.2$.

$$z_{ij} = \frac{x_{ij}}{max(i)} \tag{1}$$

Finally, it is necessary to indicate the match between a developer d_j and a task x_j. For this purpose, we calculate the similarity between the vectors $FV_{x_{a_{i_{d_j}}}}$ and $FV_{x_{a_i}}$ given the Manhattan similarity metric (Equation 2), where u and v are the target vectors and k is their size. This metric is derived from Manhattan Distance, in which the maximum distance between two vectors depends on their size. For instance, the distance between two five-size vectors, when the first is composed only of zeroes and the second only of ones, the distance is 5 (five). Similarly, if the vectors are size 10 (ten), the distance is 10 (ten). Since the FVs can have different sizes, depending on the structured task, we decided to use the Manhattan similarity, which provides values in the range [0, 1], regardless of size. Moreover, we chose this metric because it provided the best results in our experiments (see Section IV).

$$Manhattan\ similarity = 1 - \frac{\sum_{i=1}^{k} |u[i] - v[i]|}{k} \tag{2}$$

2) DEVELOPER SKILL QUANTIFICATION

Apart from the developer's knowledge, we also use his skills for forming teams. To quantify the developer's skills, we used the number of times he implemented a given structured task with a specific standard description. Since the tasks are categorized, it is possible to retrieve and reuse them independent of the project. As in the previous step, we normalized the values of the standard task descriptions for each developer, and the process is performed analogously.

3) COMPETENCE MATRIX

To map the technical competence between developer $d_j \in D$ and project $a_i \in A$, we calculated a Competence Matrix (M_{CM}). The M_{CM} is a NxM matrix, in which the n lines represent the developers available for allocation; the m columns represent the target projects, and the elements C_{ij} correspond to the developer's competence concerning the target project. Each element of the matrix M_{CM} is calculated from Equation 3, which provides a weighted score based on the quantified knowledge and skills for each developer $d_j \in D$. To prevent getting zero values from the equation when a developer does not have specific knowledge or skills, we summed ones ($1s$) in each part of the numerator (separated by the multiplication sign). The components of Equation 3 are described in what follows.

$$M_{CM} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & \dots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \dots & C_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \dots & C_{nm} \end{bmatrix}$$
$$C_{i,j} = \frac{\sum_{k=1}^{n}((1 + sim(FV_{x_{a_{i_{d_j}}}}, FV_{x_{a_i}})) \times (1 + I_{(d,t_k)}))}{n} \tag{3}$$

i represents the i^{th} developer and $i = \{1, 2, 3, \dots, n\}$; j represents the j^{th} target project and $j = \{1, 2, 3, \dots, m\}$; $sim(FV_{x_{a_{i_{d_j}}}}, FV_{x_{a_i}})$ is the similarity score between a developer FV and a task FV; $I_{(d,t_k)}$ represents the number of times the developer implemented a structured task with a determined standard description k; and n is the number of structure tasks of the sprint of the target project.

4) ALLOCATION USING GENETIC ALGORITHM

Given that we have the M_{CM} matrix, the next step is to allocate the developers to the projects. Let T be a mapping (i.e., an allocation) of a developer d_j to a_i given by $T : D \rightarrow A$. Hence, T is the set of formed teams $T = \{(t_1, a_1), (t_2, a_2), \dots, (t_m, a_m)\}$, where t_j contains a set of developers allocated to the project a_j. Our goal is to define T such that each team of the developers t_i mapped to each project a_i maximizes the possible knowledge for each $a_i \in A$.

For this purpose, we modeled the problem as a search and optimization one. According to Harman *et al.* [29], Random search, Hill Climbing, Simulated Annealing, and Genetic Algorithm are search techniques widely used to solve software engineering problems. By analyzing such techniques, we concluded that GA has a good fit due to (i) its robustness for local optima, (i) flexibility to define domain-driven heuristics (i.e., fitness functions), and suitability to solve multi-objective search problems [54]–[56]. For this last point, even though we modeled the problem only to maximize the technical competences for each project, it would be easier to extend to, for instance, also minimize costs.

The proposed solution uses a permutation chromosome that allows representing the solutions through the combination of elements. Figure 3 presents an example of the structure. The indexes represent the spots to be filled for the organization's projects. Each index corresponds to a gene that can assume an integer value that represents a developer identifier. For instance, Figure 3 (a) shows the chromosome structure for a organization with three target projects and 12 available developers. The indexes of 0 to 4 represent the spots to be filled for the *Project A*. Similarly, we have the indexes of 5 to 7 for the *Project B* and 8 to 11 *Project C*. Each gene can assume an integer value from 1 to 12, which represent unique identifiers for developers. Figure 3 (b) shows a a candidate solution. In this case, we have *Project A* $= 05, 02, 09, 11, 01$; that is, the team of this project would

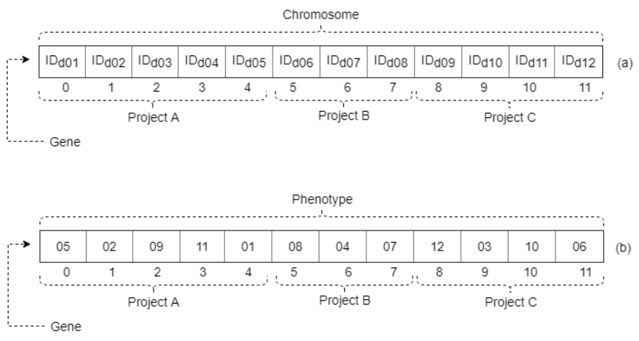


FIGURE 3. Example of a chromosome structure.

be formed by developers whose *ids* correspond to the set numbers. Analogously, we have that $Project\ B = 08, 04, 07$ and $Project\ C = 12, 03, 10, 06$.

5) FITNESS FUNCTIONS

The fitness function is a fundamental GA mechanism because it measures how close a candidate solution is to an optimal one [57]. When this function is not representative, good and bad solutions are considered similar, which can result in discarding an optimal solution during the algorithm execution. In this study, we propose two fitness functions to form agile teams.

Our first fitness function, $f_1(X)$ (Equation 4), is called *Redundancy Allocation*. In software development, the concept of redundancy reflects a level of overlap or excess capacity in expertise, which is essential for effective collaboration and coordinated action [58]. Scrum teams must be cross-functional, i.e., the team must possess all the competences necessary to carry out the work without depending on others who are not part of the team. Therefore, redundant competences among team members allow them to perform (parts of) each other's jobs and replace each other as circumstances demand [59]. Thus, we designed $f_1(X)$ to search for cross-functional teams with redundancy competences. Mathematically, it calculates the average technical competence of the developers in the candidate solution, aiming to find those who have the highest score as possible. Thus, the greater the number of competences possessed by the developer that is demanded by the target project, the higher his technical competence score will be.

$$f_1(X) = \frac{\sum_{i=1}^{n} C(d_i, p_i)}{n} \tag{4}$$

In $f_1(X)$, X is the chromosome to be evaluated; n is the number of genes of the chromosome; i is the gene position in the chromosome structure; and $C(d_i, p_i)$ is the technical competence (Section III-B3) between the developer in index i and the project in same index.

Our second fitness function is the *Specialized Allocation*, which is represented by $f_2(X)$ (Equation 5). It allows for focusing on the knowledge related to specific technologies. The main difference from $f_1(X)$ is with regards to the developer knowledge quantification step (see Section III-B1). By default, during the generation of developer and task FVs, all tags may assume values (after the normalization process) from the range $[0, 1]$. However, in this case, the PM can define specific weights for the tags to represent specialized knowledge. For instance, a PM may desire a cross-functional team in which some tasks require knowledge in Angular, Typescript, HTML/CSS, and MySQL. However, he may want specialized knowledge of Typescript and MySQL. Thus, he can define a multiplication factor to increase the weight of the chosen tags. In this case, each corresponding tag value in developer and task FVs will no longer be in the range $[0, 1]$. Instead, it will assume the new range $[0, x5]$. This modification impacts the similarity calculation between FVs; thus, developers who possess knowledge related to the weighted tags will have a higher technical competence score ($C'(D_i, P_i)$). It is important to highlight that, even though $f_2(X)$ emphasizes the knowledge related to specific technologies, it still considers the remain tags of the structured tasks, preserving the cross-functional characteristic.

$$f_2(X) = \frac{\sum_{i=1}^{n} C'(D_i, P_i)}{n} \tag{5}$$

IV. VALIDATION

This section presents details regarding the validation of the proposed solution. Section IV-A details the construction of the dataset used to validate our approach. Section IV-B describes the process to define the parameter settings of the solution. Finally, Section IV-C presents the scenarios used to validate our solution.

A. DATASET CONSTRUCTION

To build the dataset, we collected data from a Brazilian software organization. The organization granted access to its repositories, and we were able to collect data from specific software projects. As a result, we gathered data from 12 Scrum projects executed between 2015 and 2018. On average, each project was composed of eight sprints, 28 user stories, and 124 tasks. All sprints lasted for fifteen days. The overall dataset is composed of 1496 tasks, which were mapped into 1063 structured tasks, implemented by 52 different developers. Also, for confidentiality reasons, the projects' and developers' names were anonymized. Table 1 presents details about the dataset.

1) APPLICATION OF THE STRUCTURED TASK MODEL

After collecting the data, we applied the STM (Section III-A) to structure the tasks. We designed the STM to be applied during the development cycle of the projects. However, we could not accomplish this because we used data from past projects. Before starting to apply the model, we elicited the technologies of all projects to generate a tag repository for easing assigning tags. By establishing predefined tags, we prevented from having two different tags to represent the same technology.

TABLE 1. Dataset statistics.

Project ID	Number of sprints	Number of user stories	Number of technical tasks	Number of structured tasks
P01	9	43	264	212
P02	7	22	128	112
P03	4	45	111	93
P04	8	21	105	88
P05	7	12	77	45
P06	6	20	77	61
P07	10	33	135	64
P08	3	8	71	53
P09	6	32	136	107
P10	12	69	98	48
P11	13	16	164	63
P12	14	19	130	117

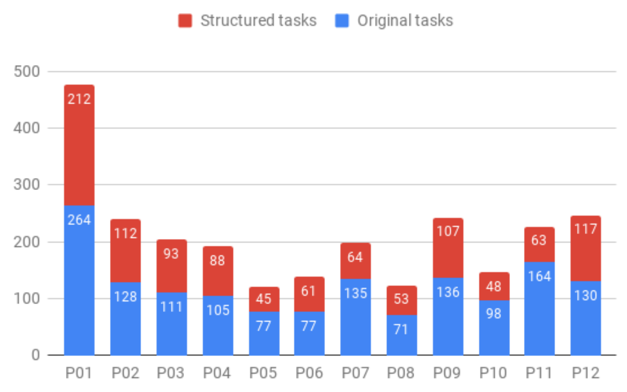


FIGURE 4. Original tasks and structured tasks of each project.

We applied STM following two phases. First, we conducted the TLP (Section III-A) with the original project teams. All teams were composed of half or more members from their original formation. Before starting, we explained the process for each team separately. It is essential to mention that during this process, we instructed the developers to search for the tags in our repository, and if they did not find any suitable options, they could define new tags. In the end, the new tags were checked, adjusted, and added to the repository. Overall, 45 distinct tags were used to represent the projects' technologies. Out of the 12 projects, eight ($P01$, $P02$, $P03$, $P08$, $P09$, $P10$, $P11$ and $P12$) were focused on developing web products, two ($P4$ and $P05$) focused on developing mobile (Android) products, and the remaining ($P06$ and $P07$) developed hybrids products (web and mobile).

We performed the TCP (Section III-A) during the second phase. For this purpose, we used the companies' standard task descriptions, which are made available in our supplementary material. These descriptions represent only technical tasks, i.e., tasks created to develop a product feature. We did not map tasks that represent bug fixes, tests, studies, and others. The process of creating the standard task descriptions is out of the scope of this paper.

To apply the TCP, we presented the descriptions to the teams and instructed them to map each original task of the project into a corresponding standard task description. Figure 4 presents the number of structured tasks in comparison to the number of original tasks of each project. Out of the 1496 original tasks of the dataset (Section IV-A), the teams managed to map 1063 tasks into structured tasks.

B. PARAMETER SETTINGS

The performance of the GAs depends on their parameter configurations. However, there is no standard configuration that can provide optimal results regardless of the domain. There are recommended values in the literature that can serve as a start point, but an optimal configuration can only be determined through experimentation. Thus, we performed an empirical study to select the optimal setting for the GA parameters and the distance measure. Besides, we validated the fitness functions of the GA. Table 2 shows the tested parameters which represent common choices in the related works. There were 1728 different configurations to be tested.

The proposed solution uses a permutation chromosome that demands specific methods for crossover and mutation. We used the Jenetics library [60] as the base for the implementation of the GA. Jenetics is a popular tool for developing Evolutionary Algorithms and Genetic Programming solutions.

Regarding the distance measure, we chose to evaluate the Euclidean, Manhattan, Canberra, and Chebyshev distances, since they are popular metrics to calculate the proximity between two points. These metrics are implemented in the Apache Commons Mathematics [2] library, which is a tool made up of mathematical and statistical components written in Java language.

C. VALIDATION SCENARIOS

We created seven scenarios to validate the proposed solution. These scenarios are based on data from four projects ($P01$, $P03$, $P04$ and $P05$). Two of them were web projects ($P01$ and $P03$), and the others were mobile projects ($P04$ and $P05$). The developers $D11$, $D49$, $D14$, $D42$, $D51$, $D52$ worked in project $P01$, and $D17$, $D35$, $D47$ worked in project $P4$. For each scenario, we defined the data of $P01$ and $P04$ as training data to generate the developers' profiles and the data of $P03$ and $P05$ as testing data to generate the target projects' profiles.

Additionally, we defined an expected output represented by an optimal team formation for each scenario. We selected the team members based on the analyses of the available developers' profiles and the target projects' profiles. We provided in the supplementary material a graphical view of the profiles and a further description of the validation scenarios. Table 3 shows the expected outputs which were assessed using Precision (Equation 6). We calculated the Precision from the number of relevant members of the team (defined by the authors) who are also recommended by the proposed solution and divided it by the number of recommended members.

[2]http://commons.apache.org/proper/commons-math/

TABLE 2. Tested parameters.

Distance measure	Population size	Stable Generations	Crossover method	Crossover rate	Mutation method	Mutation rate	Selection method	Survivor selection
Canberra	100	5	Partially Matched Crossover	50%	Swap Mutator	5%	Linear Classification	With Elitism
Chebyshev	500	10	-	70%	-	10%	Roulette Wheel	Without Elitism
Euclidean	1000	-	-	90%	-	20%	Stochastic universal sampling	-
Manhattan	-	-	-	-	-	-	Tournament	-

TABLE 3. Expected outputs for the validation scenarios.

Scenario	Expected output	Precision
Scenario 01V	P3 = {D11,D14,D42,D49, D51,D52} and P5 = {D17,D35,D47}	99.98%
Scenario 02V	P3 = [D11,D49,D52} and P5 = {D17,D35,D47}	98.58%
Scenario 03V	P3 = {D11,D14,D42,D49, D51,D52}	99.94%
Scenario 04V	P5 = {D35,D47,D52}	25.00%
Scenario 05V	P3 = {D11,D14,D52} and P5 = {D42,D47,D49}	91.19%
Scenario 06V	P5 = {D42,D49,D52}	99.96%
Scenario 07V	P3 = {D11,D14,D52}	99.99%

TABLE 4. Parameter settings of the proposed solution.

Parameter	Value
Population size	100
Stable generations	5
Crossover method	*Partially Mapped Crossover*
Crossover rate	50%
Mutation method	*Swap Mutator*
Mutation rate	5%
Selections method	Tournament (n = 3)
Survivor selections	Elitism (e = 1)

In scenarios $01V$, $02V$, $03V$, and $04V$, we used the *Redundancy Allocation* to provide the teams. In scenarios $05V$, $06V$, and $07V$, we used the *Specialized Allocation*.

$$Precision = \frac{|\{Rel.\ members\} \cap \{Rec.\ members\}|}{|\{Rec.\ members\}|} \tag{6}$$

Since GA is not deterministic, i.e., different executions of the algorithm can provide different results, we decided to execute each configuration 30 times per scenario. The Central Limit Theorem (CLT) states that the distribution of sample means approximates a normal distribution, as the sample size increases, regardless of that variable's distribution in the population. This fact holds especially true for sample sizes greater or equal to 30. Table 3 shows the percentage of configurations that reached maximum Precision, which was the majority of them. Thus, to choose the best one, we had to select another metric, the runtime execution. Table 4 presents the parameter settings of our solution.

V. EVALUATION

This section presents the evaluation of the proposed solution. Section V-A describes the evaluation method. Section V-B presents and discusses the evaluation results. As mentioned before, we applied a genetic algorithm to support the multiple team formation process. Therefore, the objective of the evaluation was to determine if the proposed solution provides teams capable of matching the PMs' expectations.

A. EVALUATION METHOD

We evaluated our approach with the assistance of four PMs of the organization, who have participated in projects that are part of the dataset. We interviewed the PM and constructed their professional profiles based on their years of experience and the number of projects in which they worked. Table 5 shows the PMs' profiles.

The evaluation was carried out in two phases. First, we delivered for each PM, 15 randomly chosen developer profiles, and another one representing the project in which he managed. A graphical view of the profiles is available in the supplementary material. All target projects' profiles were based on their first sprint.

Then, we asked the PMs to form teams composed of five developers, considering only the available developers' technical competences. This phase was divided into two scenarios. In *Scenario 01E*, we oriented the PMs to form their teams using the *Redundancy Allocation*. In *Scenario 02E*, they had to choose their members based on the *Expert Allocation*. To avoid biases, the PMs were not aware of the identities of the developers. By knowing who the stakeholders were, they could fail to consider only the technical competences, and they could end up selecting the team members using subjective aspects, which would compromise the evaluation.

In this phase, the PMs could choose the members without worrying about sharing organization's resources (developers). The sizes of the formed teams were chosen according to the sizes of the original project teams. We decided to restrict the number of available developers to three times the size of the teams, since the main goal was to verify if the proposed solution could provide teams similar to those provided by the PMs.

In the second phase of the evaluation, the managers were presented to the same project profiles. However, there were 52 developers available for allocation. In this case, the organization's resources had to be shared, i.e., a developer could not be part of two teams simultaneously. The aim was to simulate a multiple team formation scenario where PMs had to negotiate with each other to form their teams. Then, we presented

TABLE 5. Project managers' profiles.

Project type	Project Manager 01		Project Manager 02		Project Manager 03		Project Manager 04	
	Experience time	Number of projects	Experience time	Number of projects	Experience time	Number of projects	Experience time	Number of projects
Web project	From 2 to less than 4 years	5 to 7 projects	From 4 to less than 6 years	8 or more projects	Less than 2 years	1 project	6 years or more	5 to 7 projects
Mobile project	From 2 to less than 4 years	2 to 4 projects	Less than 2 years	1 project	No experience	None	From 2 to less than 4 years	5 to 7 projects
Scrum project	From 4 to less than 6 years	8 or more projects	From 2 to less than 4 years	2 to 4 projects	Less than 2 years	1 project	6 years or more	8 or more projects
General projects	From 4 to less than 6 years	8 or more projects	From 4 to less than 6 years	8 or more projects	Less than 2 years	1 project	6 years or more	8 or more projects

TABLE 6. Evaluation results of the first phase.

PM	Scenario	PM team	Proposed solution team	Precision
PM01	Scenario 01E	{47,D49,D40, D05,D01}	{47,D49,D40, D05,D01}	100%
PM01	Scenario 02E	{47,D49,D40, D05,**D01**}	{47,D49,D40, D05,**D23**}	80%
PM02	Scenario 01E	{41,D28,D42, D04,D40}	{41,D28,D42, D04,D40}	100%
PM02	Scenario 02E	{41,D28,**D42**, D04,D40}	{41,D28,**D12**, D04,D40}	80%
PM03	Scenario 01E	{14,D51,D39, **D37**,D46}	{14,D51,D39, **D10**,D46}	80%
PM03	Scenario 02E	{39,D51,D14, D37,**D46**}	{39,D51,D14, D37,**D18**}	80%
PM04	Scenario 01E	{52,D10,**D01**, D43,D23}	{52,D10,**D11**, D43,D23}	80%
PM04	Scenario 02E	{52,D10,**D01**, D43,D23}	{52,D10,**D07**, D43,D23}	80%
	Overall			85%

the teams provided by the proposed solution to verify if the PMs were willing to accept the recommendation.

Afterward, we sent a questionnaire to each PM to collect feedback about the teams provided by the proposed solution. They had to indicate, regarding the scenarios in which the proposed solution was not able to achieve maximum Precision, using a Likert scale (Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree), if they would agree with the recommendation of members that diverged from their original choice. Also, we asked the PMs to justify their reasons.

B. EVALUATION RESULTS

Table 6 presents the results of the first phase of the evaluation. We compared the teams provided by the PMs and the ones provided by the proposed solution, using Precision. We highlighted (bold) the divergences between the members.

Regarding $PM01$, there was a divergence in Scenario 02E with the developers $D01$ and $D23$. $PM01$ analyzed $D23$'s profile and considered him also an appropriate choice for the team. Besides, the manager said that he would agree to replace $D01$ with $D23$. There was another divergence in Scenario 02E with $PM02$. The manager agreed with the recommendation since he considers that $D42$ and $D12$ have similar competences.

Regarding $PM03$, there were divergences in both Scenarios (01E and 02E). In the first scenario, the manager disagreed with the recommendation of $D10$, because $D37$ has specific competences for the target project. For instance, $D37$ knows Cytoscape, which is a library used in most tasks of the project. The manager complemented that $D10$ does not have enough experience related to the main tasks of the projects. About Scenario 02E, $PM03$ agreed with the recommendation. Despite selecting $D46$, who has more overall knowledge, the manager said that the knowledge of $D18$ is especially required by the project.

TABLE 7. Conflict rates registered during multiple team formation process.

PM	Target project	Team	Conflict Rates P7	P8	P9	P11
PM01	P7	{52,D49,D11,D42,D05}	-	100%	60%	40%
PM02	P8	{52,D49,D11,D42,D05}	100%	-	60%	40%
PM03	P9	{52,D49,D14,D42,D47}	60%	60%	-	40%
PM04	P11	{52,D11,D14,D07,D37}	40%	40%	40%	-
	Average		67%	67%	53%	40%

There were divergences with $PM04$ in the two Scenarios. In Scenario 01E, the manager agreed with the suggestions, since $D52$ has all the necessary competences for the project, the manager selected $D01$ for the project because the developer has more knowledge about Node than $D11$. $D52$ would be able to do front-end tasks, along with $D43$ and $D23$, while $D01$ would do back-end tasks. Despite this, the manager also considered $D11$ as a valid option, since the developer would also be able to fulfill the back-end tasks with Node and MongoDB satisfactorily. In scenario 02E, $PM04$ disagreed with the suggestion, stating that he would not trade $D01$ for $D07$ since the latter has fewer skills and only on front-end tasks.

As a result, the proposed solution achieved 85% of Precision. Besides, we registered a 67% of acceptance in the scenarios that there were divergences, i.e., scenarios that the PMs agreed with the recommendation.

Table 7 shows the conflict rates during the multiple team formation process in Scenario 03E. For instance, a 100% of conflict rate means that two managers chose the same team members for their projects. In this scenario, the proposed solution achieved 75% of acceptance.

VI. THREATS TO VALIDITY

This section analyzes the threats that may affect the results of this study. We used the classification provided by Wohlin *et al.* [61].

- **Threat to internal validity:** The Structured Task model was designed to be used during the ongoing development cycle of the projects. However, some projects were already finished by the time we applied the model. Thus, we had to retrieve this data from the organization's repository. Not applying the model in the proper period may result in tasks not accurately labeled since the original team may not be available anymore, and the ones who are may not completely remember the technologies associated with a specific task. To mitigate this threat, we adopted a peer-reviewed process, in which each developer labeled his tasks, and another member from the same team reviewed the applied tags. In case of disagreement, the entire team discussed the classification.

- **Threat to external validity:** Software organizations develop products to several platforms, e.g., embedded, desktop, mobile, and web. However, our dataset is composed of mobile and web projects only, which not embrace the characteristics of all platforms, and consequently, compromise the generalization of our findings. To mitigate such a threat, we intend to enlarge the dataset, seeking to add projects from different platforms, domains, and technologies. Besides, we plan to include developers with more diversified competences.
- **Threat to construct validity:** The final configuration of the proposed solution resulted from the experimentation of similarity metrics and GA parameters. We perceive there are other alternatives besides the experimented ones, although increasing the factors' levels would exponentially increase the number of possible configurations, making the experiment more costly. To mitigate this threat, we opted for most recurrent metrics and parameters in the literature, since, in general, they provide better results.
- **Threat to conclusion validity:** Although the proposed solution was evaluated in several scenarios, which were created from real-world data and with expert support, it is possible that they do not reflect all the characteristics of a real-world environment. To mitigate this threat, we intend to improve our solution, so in the future, we will conduct a case study.

VII. CONCLUSION AND FUTURE WORK

This paper presents a genetic algorithm approach for multiple team formation in Scrum projects. Our solution introduce a systematic way to record structured data to build technical profiles during the development cycles of the projects. Additionally, our solution use the built profiles as input for the GA to search for the best global team formation in Scrum projects.

The proposed solution optimizes the MTF process by allowing the project managers to decide the formation of their teams with more reliability. Within this context, our solution could also provide training and hiring warnings. For instance, it would be possible to add constraints to the GA's fitness functions to allocate only the developers who reached a minimum score. If the size of the suggested team is not the expected one, it would be a sign that the organization needs to train its human resources or hire new ones.

To validate the proposed solution, we created seven simulated scenarios using the historical data from a repository of a Brazilian software organization. For each scenario, we defined an expected output, and the teams provided by the proposed solution were assessed using Precision. This procedure allowed us to determine an optimal parameter setting for the solution and to assess the fitness functions of the GA.

The evaluation process was divided into two phases. In the first, project managers who have worked in projects that are part of the dataset, formed teams for two different scenarios. Afterward, their teams were compared with the ones provided by our solution for the same scenarios. As a result, our solution achieved 85% of Precision. In the second phase, the PMs had to share the resources to form their teams. As a result, the proposed solution assisted them in reaching a consensus with an acceptance rate of 75%.

Our study has some limitations discussed as follows. The proposed approach uses the data from the ongoing projects of the organization to build the technical profiles. Naturally, in a starting scenario in which no previous project data is available, the proposed solution would suffer the cold start problem [62]. It would be possible to overcome this limitation by using external knowledge sources. Another limitation refers to the complexity of the tasks. We assume that tasks with the same standard text description and tags require the same effort, and consequently, are threaded as the same. To be more assertively, it would be necessary to registry the effort to implement the task. However, effort estimation of technical tasks is not a common practice in ASD.

As future work, we intend to continue the research and improve the proposed solution to minimize some limitations. We will investigate the possibility of using other project assets to enhance the developers' profiles. For instance, code snippets from task commits could be used to infer the quality of the developer's work. Additionally, external knowledge sources such as code repositories (GitHub and Bitbucket), social network (LinkedIn), and online communities (Stack Overflow and Quora) could be used to improve the profiles. Furthermore, we intend to investigate the costs and benefits of adding other types of attributes (personality traits, social interactions, interpersonal factors, and others) to complement the developers' profiles.

REFERENCES

[1] K. Araszkiewicz, "Application of critical chain management in construction projects schedules in a multi-project environment: A case study," *Proc. Eng.*, vol. 182, pp. 33–41, Jan. 2017.

[2] J. H. Gutiérrez, C. A. Astudillo, P. Ballesteros-Pérez, D. Mora-Melià, and A. Candia-Véjar, "The multiple team formation problem using sociometry," *Comput. Oper. Res.*, vol. 75, pp. 150–162, Nov. 2016.

[3] M. Blatter, S. Muehlemann, S. Schenker, and S. C. Wolter, "Hiring costs for skilled workers and the supply of firm-provided training," *Oxford Econ. Papers*, vol. 68, no. 1, pp. 238–257, Jan. 2016, doi: 10.1093/oep/gpv050.

[4] A. Roy, S. Sural, A. K. Majumdar, J. Vaidya, and V. Atluri, "On optimal employee assignment in constrained role-based access control systems," *ACM Trans. Manage. Inf. Syst.*, vol. 7, no. 4, pp. 1–24, Jan. 2017.

[5] E. Dantas, A. Costa, M. Vinicius, M. Perkusich, H. Almeida, and A. Perkusich, ''An effort estimation support tool for agile software development: An empirical evaluation,'' in *Proc. Int. Conf. Softw. Eng. Knowl. Eng.*, Jul. 2019, pp. 82–87.

[6] F. Ramos, A. Pedro, M. Cesar, A. Costa, M. Perkusich, H. Almeida, and A. Perkusich, ''Evaluating software developers' acceptance of a tool for supporting agile non-functional requirement elicitation,'' in *Proc. Int. Conf. Softw. Eng. Knowl. Eng.*, Jul. 2019, pp. 26–31.

[7] A. Costa, F. Ramos, M. Perkusich, E. Dantas, E. Dilorenzo, F. Chagas, A. Meireles, D. Albuquerque, L. Silva, H. Almeida, and A. Perkusich, ''Team formation in software engineering: A systematic mapping study,'' *IEEE Access*, vol. 8, pp. 145687–145712, 2020.

[8] M. F. Abrar, M. S. Khan, S. Ali, U. Ali, M. F. Majeed, A. Ali, B. Amin, and N. Rasheed, ''Motivators for large-scale agile adoption from management perspective: A systematic literature review,'' *IEEE Access*, vol. 7, pp. 22660–22674, 2019.

[9] P. Jain, A. Sharma, and L. Ahuja, ''The impact of agile software development process on the quality of software product,'' in *Proc. 7th Int. Conf. Rel., INFOCOM Technol. Optim. (Trends Future Directions) (ICRITO)*, Aug. 2018, pp. 812–815.

[10] S. Gupta and D. Gouttam, ''Towards changing the paradigm of software development in software industries: An emergence of agile software development,'' in *Proc. IEEE Int. Conf. Smart Technol. Manage. Comput., Commun., Controls, Energy Mater. (ICSTM)*, Aug. 2017, pp. 18–21.

[11] H. Cornide-Reyes, F. Riquelme, R. Noel, R. Villarroel, C. Cechinel, P. Letelier, and R. Munoz, ''Key skills to work with agile frameworks in software engineering: Chilean perspectives,'' *IEEE Access*, vol. 9, pp. 84724–84738, 2021.

[12] R. Hoda, N. Salleh, and J. Grundy, ''The rise and evolution of agile software development,'' *IEEE Softw.*, vol. 35, no. 5, pp. 58–63, Sep. 2018.

[13] VersionOne. (2019). *13th Annual State of Agile Development Survey Results*. Accessed: Jan. 7, 2019. [Online]. Available: https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-re%port/473508

[14] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed, Portable Documents*. Reading, MA, USA: Addison-Wesley, 2003.

[15] R. Elamin and R. Osman, ''Towards requirements reuse by implementing traceability in agile development,'' in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2017, pp. 431–436.

[16] Y. Wu, X. Xie, J. Wang, D. Deng, H. Liang, H. Zhang, W. Chen, and S. Cheng, ''Forvizor: Visualizing spatio-temporal team formations in soccer,'' *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 65–75, Jan. 2019.

[17] E. Alhazmi, S. Horawalavithana, J. Skvoretz, J. Blackburn, and A. Iamnitchi, ''An empirical study on team formation in online games,'' in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Jul. 2017, pp. 431–438.

[18] K. He, Z. Liang, T. Cui, Z. Ke, Z. Liu, Q. Zhao, and F. Fang, ''Formation optimization of RoboCup3D soccer robots using Delaunay triangulation network,'' in *Proc. Chin. Control Decis. Conf. (CCDC)*, Jun. 2018, pp. 224–229.

[19] L. Feola and V. Trianni, ''Adaptive strategies for team formation in minimalist robot swarms,'' *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4079–4085, Apr. 2022.

[20] L. Zhang, T. Song, Y. Tong, Z. Zhou, D. Li, W. Ai, L. Zhang, G. Wu, Y. Liu, and J. Ye, ''Recommendation-based team formation for on-demand taxi-calling platforms,'' in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 59–68, doi: 10.1145/3357384.3357869.

[21] L. Liang, X. Cheng, and T. Ikenaga, ''Team formation mapping and sequential ball motion state based event recognition for automatic data volley,'' in *Proc. 16th Int. Conf. Mach. Vis. Appl. (MVA)*, May 2019, pp. 1–4.

[22] G. Barnabò, A. Fazzone, S. Leonardi, and C. Schwiegelshohn, ''Algorithms for fair team formation in online labour marketplaces,'' in *Proc. Companion Proc. World Wide Web Conf.*, May 2019, pp. 484–490, doi: 10.1145/3308560.3317587.

[23] V. M, S. Salimath, A. S. Shettar, and G. Bhadri, ''A study of team formation strategies and their impact on individual Student learning using educational data mining (EDM),'' in *Proc. IEEE 10th Int. Conf. Technol. Educ. (T4E)*, Dec. 2018, pp. 182–185.

[24] S. A. Licorish, M. Galster, G. M. Kapitsaki, and A. Tahir, ''Understanding students' software development projects: Effort, performance, satisfaction, skills and their relation to the adequacy of outcomes developed,'' *J. Syst. Softw.*, vol. 186, Apr. 2022, Art. no. 111156.

[25] V. Isomöttönen and E. Ritvos, ''Digging into group establishment: Intervention design and evaluation,'' *J. Syst. Softw.*, vol. 178, Aug. 2021, Art. no. 110974.

[26] L. C. 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.

[27] 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.

[28] C. Crawford, Z. Rahaman, and S. Sen, ''Evaluating the efficiency of robust team formation algorithms,'' in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, Cham, Switzerland: Springer, 2016, pp. 14–29.

[29] M. Harman, S. A. Mansouri, and Y. Zhang, ''Search-based software engineering: Trends, techniques and applications,'' *ACM Comput. Surv.*, vol. 45, no. 1, p. 11, 2012.

[30] 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. Int. Conf. Softw. Eng. Knowl. Eng.*, Jul. 2018, pp. 1–6.

[31] 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.

[32] 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.

[33] 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.

[34] V. S. Baghel and S. D. Bhavani, ''Multiple team formation using an evolutionary approach,'' in *Proc. 11th Int. Conf. Contemp. Comput. (IC)*, Aug. 2018, pp. 1–6.

[35] 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.

[36] 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.

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

[38] 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. 1545–1560, Nov. 2016.

[39] 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.

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

[41] L. Ye, H. Sun, X. Wang, and J. Wang, ''Personalized teammate recommendation for crowdsourced software developers,'' in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, Sep. 2018, pp. 808–813, doi: 10.1145/3238147.3240472.

[42] J. Zhang, P. S. Yu, and Y. Lv, ''Enterprise employee training via project team formation,'' in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Feb. 2017, pp. 3–12.

[43] T.-L. 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.

[44] E. Dilorenzo, E. Dantas, M. Perkusich, F. Ramos, A. Costa, D. Albuquerque, H. Almeida, and A. Perkusich, ''Enabling the reuse of software development assets through a taxonomy for user stories,'' *IEEE Access*, vol. 8, pp. 107285–107300, 2020.

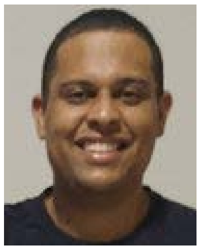
[45] R. Shankarmani, S. S. Mantha, V. Babu, D. Mehta, K. Khatri, and P. Kaushil, ''A decision support system utilizing a semantic agent,'' in *Proc. IEEE Int. Conf. Softw. Eng. Service Sci.*, Jul. 2010, pp. 442–447.

[46] M. Fazel-Zarandi and M. S. Fox, "An ontology for skill and competency management," in *Proc. FOIS*, 2012, pp. 89–102.
[47] I. K. Raharjana, D. Siahaan, and C. Fatichah, "User stories and natural language processing: A systematic literature review," *IEEE Access*, vol. 9, pp. 53811–53826, 2021.
[48] A. R. Amna and G. Poels, "Systematic literature mapping of user story research," *IEEE Access*, vol. 10, pp. 51723–51746, 2022.
[49] D. Wang, J. Su, and H. Yu, "Feature extraction and analysis of natural language processing for deep learning english language," *IEEE Access*, vol. 8, pp. 46335–46345, 2020.
[50] T. Al-Moslmi, M. Gallofre Ocana, A. L. Opdahl, and C. Veres, "Named entity extraction for knowledge graphs: A literature overview," *IEEE Access*, vol. 8, pp. 32862–32881, 2020.
[51] S. Singh and A. Mahmood, "The NLP cookbook: Modern recipes for transformer based deep learning architectures," *IEEE Access*, vol. 9, pp. 68675–68702, 2021.
[52] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Extracting conceptual models from user stories with visual narrator," *Requirements Eng.*, vol. 22, no. 3, pp. 339–358, Sep. 2017.
[53] M. Zanoni, F. A. Fontana, and F. Stella, "On applying machine learning techniques for design pattern detection," *J. Syst. Softw.*, vol. 103, pp. 102–117, May 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215000217
[54] A. Delgoshaei, A. Ali, M. K. A. Ariffin, and C. Gomes, "A multi-period scheduling of dynamic cellular manufacturing systems in the presence of cost uncertainty," *Comput. Ind. Eng.*, vol. 100, pp. 110–132, Oct. 2016.
[55] S. Kardani-Moghaddam, F. Khodadadi, R. Entezari-Maleki, and A. Movaghar, "A hybrid genetic algorithm and variable neighborhood search for task scheduling problem in grid environment," *Proc. Eng.*, vol. 29, pp. 3808–3814, Jan. 2012.
[56] A. Delgoshaei, M. K. M. Ariffin, B. T. H. T. B. Baharudin, and Z. Leman, "Minimizing makespan of a resource-constrained scheduling problem: A hybrid greedy and genetic algorithms," *Int. J. Ind. Eng. Comput.*, vol. 6, no. 4, pp. 503–520, 2015.
[57] J. Liu, X.-G. Luo, X.-M. Zhang, F. Zhang, and B.-N. Li, "Job scheduling model for cloud computing based on multi-objective genetic algorithm," *Int. J. Comput. Sci.*, vol. 10, no. 1, p. 134, 2013.
[58] T. E. Fægri, T. Dybå, and T. Dingsøyr, "Introducing knowledge redundancy practice in software development: Experiences with job rotation in support work," *Inf. Softw. Technol.*, vol. 52, no. 10, pp. 1118–1132, Oct. 2010.
[59] N. B. Moe, T. Dingsøyr, and E. A. Røyrvik, "Putting agile teamwork to the test—An preliminary instrument for empirically assessing and improving agile software development," in *Proc. Int. Conf. Agile Processes Extreme Program. Softw. Eng.* Cham, Switzerland: Springer, 2009, pp. 114–123.
[60] F. Wilhelmsttter, "Jenetics–Java genetic algorithm," Jenetics, Vienna, Austria, Tech. Rep., 2012.
[61] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Cham, Switzerland: Springer, 2012.
[62] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing cold-start problem in recommendation systems," in *Proc. 2nd Int. Conf. Ubiquitous Inf. Manage. Commun. (ICUIMC)*, 2008, pp. 208–211, doi: 10.1145/1352793.1352837.

ALEXANDRE COSTA received the M.Sc. and Ph.D. degrees in computer science from the Federal University of Campina Grande, Paraiba, Brazil, in 2014 and 2019, respectively. He has been a Professor with the Federal Institute of Paraiba, 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, the main topics of interest are 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, Paraiba, Brazil, in 2012 and 2019, respectively. He is currently a Professor with the Federal Institute of Paraiba. Further, he is 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 artificial intelligence applied to software engineering to solve complex problems, agile software development, and requirement engineering focused on supporting the elicitation of non-functional requirements on scrum-based projects.

MIRKO PERKUSICH received the Ph.D. degree in computer science. He is currently the Research Manager at Virtus, leading the Intelligent Software Engineering Research Group. His current research interests include applying intelligent techniques, including recommender systems, to solve complex software engineering problems.



ADEMAR DE SOUSA NETO is currently pursuing the Ph.D. degree with the Virtus Research. He is also a Researcher in software engineering at Virtus Research. In particular, seeking to improve the software engineering process using intelligent approaches, including artificial intelligence and big data.

LUIZ SILVA received the M.Sc. degree in computer science from the Federal University of Campina Grande, Brazil, in 2019. His research interests include machine learning and application of intelligent techniques to solve software engineering problems.

FELIPE CUNHA is currently pursuing the Ph.D. degree 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 interest includes artificial intelligence applied to software engineering.

THIAGO RIQUE is currently pursuing the Ph.D. degree with the Federal University of Campina Grande. He is also a Professor with the Federal Institute of Paraiba. Further, he is a member of the Intelligent Software Engineering Research Group, Virtus, which is a Research, Development, and Innovation Center in Information Technology.

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, in 2004 and 2007, respectively. He is currently the Head of the Intelligent Software Engineering Group, and the Founder and the Director of Operations with the Virtus Innovation Center (VIRTUS/UFCG). He is also a Researcher with the Embedded and Pervasive Computing Laboratory (Embed-ded/UFCG). He has been a Professor with the Computer and Systems Department, Federal University of Campina Grande (UFCG), since 2006. He is also an Executive Director of the EMBRAPII Unit, 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 and training courses for industry in the context of software engineering. He has more than 200 papers published, and 37 master's thesis and 13 doctoral dissertations advised. His current research interest includes applying intelligent techniques to solve complex software engineering problems.

ANGELO PERKUSICH (Member, IEEE) received the master's and Ph.D. degrees in electrical engineering from the Federal University of Paraíba, in 1987 and 1994, respectively. He was a Visiting Researcher with the Department of Computer Science, University of Pittsburgh, PA, USA, from 1992 to 1993, and developed research activities on software engineering and formal methods. He has been a Professor with the Electrical Engineering Department (DEE), Federal University of Campina Grande (UFCG), since 2002. 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), and private companies. He is the Founder and the Director of the Virtus Innovation Center and the Embedded and Pervasive Computing Laboratory. The 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 and training courses for industry in the context of software for real-time systems, software engineering, embedded systems, computer networks, and formal methods. His research interests include embedded systems, software engineering, mobile pervasive computing, and formal methods, with more than 300 papers published, and 80 master's thesis and 21 doctoral dissertations advised.