A Multi-criteria Approach for Team Recommendation

Michael Arias(✉), Jorge Munoz-Gama, and Marcos Sepúlveda

Department of Computer Science, School of Engineering,
Pontificia Universidad Católica de Chile, Santiago, Chile
m.arias@uc.cl, {jmun,marcos}@ing.puc.cl

Abstract. Team recommendation is a key and little-explored aspect within the area of business process management. The efficiency with which the team is conformed may influence the success of the process execution. The formation of work teams is often done manually, without a comparative analysis based on multiple criteria between the individual performance of the resources and their collective performance in different teams. In this article, we present a multi-criteria framework to allocate work teams dynamically. The framework considers four elements: (i) a resource request characterization, (ii) historical information on the process execution and expertise information, (iii) different metrics which calculate the suitability of the work teams taking into account both individual performance as well as collective performance of the resources, and (iv) a recommender system based on the Best Position Algorithm (BPA2) to obtain a ranking for the recommended work teams. A software development process was used to test the usefulness of our approach.

Keywords: Team recommendation · Resource allocation · Process mining · Business processes · Recommender systems · Organizational perspective

1 Introduction

The task of allocating resources to activities within a business process is a key aspect in the area of Business Process Management (BPM) [12,13,27]. The efficiency with which this task is performed has a high relevance in terms of resources productivity, quality, performance and cost of the process. The use of information about the resources that take part of the process activities is recommended to improve the task of allocating resources, for example, to support the discovery of patterns related to resource assignment [23]. In the context of processes that require a collaborative environment among their activities, the correct selection of work teams represents a challenge for those in charge, as it requires the selection of those resources that working together provide the best performance to execute the activities that conform a process. Despite its relevance, the formation of work teams is often done manually, without a comparative analysis based on multiple criteria between the individual performance of the resources and their collective performance in different teams. Seeking for a resolution to the problem of resource allocation, different methods have been proposed, including: association rules [13], decision trees [14], and Markov models [12]. Despite this, there is a lack of approaches that bring adequate automatic support to compose and allocate activities to work teams within the area of BPM [8]. From the literature, it is possible to identify different criteria which are used to evaluate resources in the context of work teams, such as: resource availability [16], abilities required to perform a task [11,24], task complexity [24], effort estimation [19], and sociometric techniques [4]. In [24], a genetic algorithm is proposed to solve the problem of team formation, including practical considerations: precedence among tasks, role, and competence level of the resource. In [5], an optimization-based approach to support staffing in a software project is introduced, where information related to the activities, the available human resources, and a set of restrictions established by the organization (e.g., schedule deadline, project budget, maximum allocation) are considered. Kumar et al. [15] present a model that measures compatibility among resources when assigning work items to collaborative groups. In [6], they focus on team formation within the context of agile software development methodologies, specifying rules to choose the best developers for a given project. Recently, in [17] a method to improve resource allocation in workflow management systems by computing the social relation among two resources was proposed. This approach only considers the time perspective. Cabanillas et al. [8] present an approach to select teams proposing a language named RAL-Team to describe the work teams, which is extended to establish selection conditions verified at the time of determining the resources that are assigned to the different activities of the business process. This approach does not take into account the historical information about past process executions, neither support the team composition at run-time. From the literature revised, we detected some limitations. First, there is a lack of methods that support the formation of teams at run-time. Second, there is a lack of mechanisms that support decision making through the recommendation of suitable teams, in which the historic performance of the resources, both individually and collectively, is analyzed. Third, the resources are always allocated considering one allocation unit: the activity level. Fourth, there is a need to count with multi-criteria approaches that are scalable and user-oriented, where the criteria used to determine the most suitable allocation can be extended in a faster and easier manner for the decision maker.

In this work, we extend the Framework for Recommending Resource Allocation based on Process Mining presented in [3]. We approach resource allocation considering work team recommendation. A role is a group of resources that are interchangeable in the sense that any member of the group can perform a given set of activities. It is also called a resource class [10]. For simplicity, a usual assumption is that each activity can be performed by only a single role. On the other hand, a resource can have multiple roles, e.g., a software architect can also be the project leader. Usually, in a given process, several roles can be identified. A work team (hereinafter and indistinctly, a team) is a group of resources, each playing a specific role, that together can perform a process instance. This work presents a novel approach that recommends the most suitable teams to execute a request defined at run-time. We consider sub-processes as the target allocation unit; however, it can also be used to allocate resources at the activity level or at the process level, as a whole. Inspired by workflow resource patters [22], three allocation types are considered: capability-based allocation, history-based allocation and role-based allocation. We proposed a multi-criteria approach in which various metrics are evaluated to incorporate the evaluation of the individual performance of the resources and their group performance through different criteria: fitting between resources capabilities and the expertise required to perform an activity, past performance (frequency, duration, quality and cost), and the current workload of each resource. The metrics are combined to generate a final recommendation ranking using BPA2 [2]. We can formulate the process of generating the ranking as a problem of obtaining the k most relevant items in a multi-dimensional dataset. To accomplish this goal, we propose the use of BPA2 to answering top-k queries using lists of data items sorted by their local scores. Also, we report experimental obtained results.

As a running example we use a traditional software development process: the Waterfall life cycle [21]. It consists of five phases run sequentially: (i) Requirements analysis (role: analyst), (ii) System design (role: designer), (iii) Implementation or coding (role: developer), (iv) Testing (role: tester), and (v) Operation and maintenance (role: support agent). The Waterfall is used within the context of software engineering [25] for systems development, where each phase is conducted by a resource that performs a specific role. Periodically, companies that develop software require the selection of resources to form the team that will execute each phase, in order to generate a specific software solution.

This article is organized in the following manner: Sect. 2 gives the preliminaries that are necessary for the rest of the article. Section 3 describes the method used to evaluate and recommend work teams. Section 4 focuses on the experiments performed and discusses the results obtained when validating the proposed approach. Finally, Sect. 5 presents the conclusions and future work.

2 Preliminaries

In this section we present the elements that form part of the framework to perform the recommendation of the most suitable work teams.

Resource Request Characterization

As we established before, a team is a group of resources, each playing a specific role, that together can perform a process instance. To recommend resources to form a team, we must first define the properties each member of the work team must comply with to fulfil its role in a process instance.

Definition 1 ***(Resource Request Characterization).*** *A resource request characterization $c = (f_1, \ldots, f_n)$ is a multi-factor representation of the desired request properties, where f_i is an element of a finite set.*

We used at least one of the factors to express the need for a specific role. The two-factor characterization proposed is a factor-tuple $c = (role, type)$, where *role* defines the organizational role for which the resource is being requested, and *type* is the typology of the process execution instance that requests the resource. For example, in the Waterfall life cycle, a different *role* is required for each phase (e.g., an analyst or a tester), and *type* corresponds to the project type for which the resource is requested (e.g., create a website or a mobile app). To determine the phases of the process that could require a different role, we can consider the semantics of the process itself and then perform a decomposition manually. Alternatively, this decomposition could also be done through automated approaches, such as Passages [1], o Single-Entry Single-Exit (SESE) [18].

For each resource request characterization c, we use resource allocation information that includes: historical information of previous process executions, contextual information (e.g., expertise information available in corporate information systems), and weights describing the importance of each of the allocation metrics. Our framework returns a ranking of the most suitable resources to be allocated to each request.

Resource Allocation Metrics

For our approach, we adopt the use of the six dimensions introduced in [3], where historical and contextual information is combined to evaluate the resources through different metrics in accordance with the specified criteria (dimensions). The proposed dimensions are:

– **Frequency:** measures the rate of occurrence that a resource has completed the requested characterization.
– **Performance:** measures the execution time that a resource has achieved performing the requested characterization.
– **Quality:** measures the customer evaluation of the execution of the requested characterization performed by a resource.
– **Cost:** measures the execution cost of the requested characterization performed by a resource
– **Workload:** measures the actual idle level of a resource considering the characterizations executed at the time
– **Expertise:** measures the ability level at which a resource is able to execute a characterization

We introduce a resource process cube $\mathbb{Q}$ to manage the historical information, and the expertise matrices $\mathbb{E}_r$ and $\mathbb{E}_c$ representing the expertise information. By means of the cube and the matrices, the required knowledge is generated, with the purpose of determine the suitability of the resources that take part of the recommended team. The *cube* $\mathbb{Q}$ is a semantic representation of the historical information to be analyzed. Basic OLAP operations such as slice and dice [9], allow the extraction of specific information for each request characterization.

Definition 2 ***(Resource Process Cube).*** *Let r, c and d, be a resource, a resource request characterization, and a dimension, respectively. A resource* *process cube* $\mathbb{Q}[r][c][d]$ *represents all the historical information about the resource* r *and the characterization* c *necessary to analyze the dimension* d*. Similarly, for a given dimension* d*,* $\mathbb{Q}[\ ][c][d]$ *represents the historical information about all resources for the execution of the characterization* c*, and* $\mathbb{Q}[r][\ ][d]$ *represents the information for all the characterizations performed by the resource* r*.*

Due the nature of our approach, there is a separation between the conceptual level and how data are stored, in fact, the data could be stored in different ways, e.g., in a relational database, and then computing the metrics on demand, or by first precomputing some intermediate data, and then computing the metrics.

To illustrate the definition of the metrics for each dimension, we use the *performance* dimension. Let $\mathbb{Q}[r][c][p].avg$ be an operation that returns the average duration, considering only cases in which the resource r has taken part in executing the characterization c. Let $\mathbb{Q}$[ ][c][p].min and $\mathbb{Q}$[ ][c][p].max be the minimum and maximum duration for executing the characterization c by any resource. The performance metric is defined as follows in (1):

$$Performance\_Metric(r,c) = \frac{\mathbb{Q}[\ ][c][p].max - \mathbb{Q}[r][c][p].avg}{\mathbb{Q}[\ ][c][p].max - \mathbb{Q}[\ ][c][p].min} \tag{1}$$

Following the software development process example, if we have a characterization $c_1 = (Analyst, website)$ and a resource $r_2 = Bob$, $\mathbb{Q}[r_2][c_1][p]$ allows us to access information related to the performance dimension of resource r_2 when executing the characterization c_1. $\mathbb{Q}[r_2][c_1][p].min$ and $\mathbb{Q}[r_2][c_1][p].max$ are the minimum and maximum time *Bob* needed to execute c_1. Meanwhile, $\mathbb{Q}[r_2][c_1][p].avg$ is the average time required by *Bob* to execute c_1. In the same way, $\mathbb{Q}[\ ][c_1][p].min$ and $\mathbb{Q}[\ ][c_1][p].max$ are the minimum and maximum time required considering all resources. For example, be 54 and 120 the minimum and maximum time needed to perform the characterization c_1, and 59 the average duration of *Bob*, the *Performance_Metric(r,c)* = 0,92.

At the same time, the *expertise matrices* $\mathbb{E}_r[r][e]$ and $\mathbb{E}_c[c][e]$ allow us to represent the expertise e of a resource r and the desired level of expertise required to execute a characterization c, respectively. We represent the expertise information considering the Human Resource Meta-Model (HRMM) [20], where the expertise can be organized by competencies, skills, and knowledge. To evaluate how a resource fits with the expertise required to perform a given characterization, we compare the value of each level of expertise $\mathbb{E}_r$ with the corresponding value in $\mathbb{E}_c$. This comparison allows determining how qualified a resource r is, according to an under-qualification and an over-qualification metric. For example, given the characterization $c_1 = (Analysis, website)$ the desired expertise to perform this characterization is a mid-high level on problem solving, business analysis, and communication skills, represented as [2, 2, 2], meanwhile, the resource $r1$ has a low level in the above aspects, represented as [0, 1, 0]. If the expertise of a resource $r1$ entirely match with the required expertise, the value for both expertise metrics will be 1. Similarly, we established metrics for the other dimensions. For further detail about the definition of all the different metrics used, reader can refer to the previous publication [3]. From now on, we will refer to the metrics in a generic way as $metric_j(\mathbb{Q}(L), r, c)$, being L the event log used to compute the cube.

Best Position Algorithm Problem

The challenge of recommending a group of resources to each request can be formulated as a problem of finding the best k resources in a group of ordered lists, where each list corresponds to one of the metrics considered. With the intention of giving a recommendation ranking, we were inspired in the use of portfolio-based algorithm selection [26] as a strategy to obtain the k most relevant items in a given group of data (top-k technique) considering multiple criteria. Akbarinia et al. [2] proposed two algorithms (BPA and BPA2) to process the top-k queries starting from the ordered lists of data in accordance to each criterion considered. BPA2 is a suitable algorithm to accomplish the goal of obtaining the k most relevant items in a multi-dimensional dataset, because it requires a reduced number of data accesses, a lower query response time and execution costs, with respect to other top-k query processing algorithms, such as BPA and the Threshold Algorithm (TA), due its threshold management and early stop condition.

3 Team Recommendation

To develop a team recommendation, we consider the elements presented in Sect. 2. Figure 1 shows the general framework proposed. This framework helps to specify a request at run-time, as well as weights that describe the level of importance of each criterion established by the user. The recommendation takes account of contextual and historical information. The defined metrics enable the team resources to be evaluated in accordance with the specified criteria. Notice that in real-life scenarios, the best resources, evaluated individually, are not always those who compose the best teams collectively. This may occur in the composition of teams in software development projects or consulting projects, emergency medical teams, among others. Therefore, we created a new method of evaluation to propose suitable work teams, considering both individual and group behavior, which we will present in this section.

To obtain the recommendation, we evaluate each team based on their history, expertise, and availability, considering: (a) the individual behavior of each resource that forms the team, independent of the teams it had belong to, (b) the

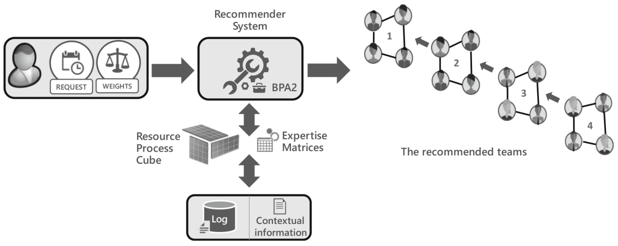

Fig. 1. General framework for team resource recommendation

behavior of each resource when it has previously formed part of the team that is being evaluated. Later, the BPA2 algorithm is used to recommend the most suitable teams. In the following, we will explain how each step is performed.

Team Filtering: A team is a group of resources that together can perform a process instance, each taking care of a specific resource request characterization (an organizational role for a specific type of process instance). To consider the history of a particular team, we extract from the event log those equivalent process instances that they have performed together in the past.

Definition 3 ***(Team Filtering).*** *Let L, $\mathbf{tc}$, $\mathbf{tr}$, be an event log, a tuple of resource request characterizations, and a tuple of resources, respectively.* Team Filtering *is a function $team\_filter(L, \mathbf{tc}, \mathbf{tr}) = L_{team}$ that returns an event log that contain all events of the log L that belong to those traces that contain only the characterizations $\mathbf{tc}$, and where each resource request characterization in $\mathbf{tc}$ has been performed by the corresponding resource in $\mathbf{tr}$.*

We will refer to $\mathbb{Q}(L)$ as the resource process cube created based on the historical information contained in an event log L. Moreover, $\mathbb{Q}(team\_filter(L, \mathbf{tc}, \mathbf{tr}))$ symbolizes the resource process cube that represents the historical information where the team $\mathbf{tr}$ has worked together performing different occurrences of the resource request characterizations $\mathbf{tc}$ associated to the process registered in L.

Individual Assessment: To determine the individual evaluation of a resource, independent of the teams it belonged, we used the *cube* $\mathbb{Q}(L)$, which represents all the historical information contained in L. Based on $\mathbb{Q}(L)$ and the expertise matrices, the different metrics are calculated; notice that all metrics are normalized between 0 and 1. For each metric, an ordered list which evaluates the behavior of each resource of the team in that dimension is created. Each metric is weighted by previously specified weights, e.g., more importance could be given to the dimensions of frequency and quality, above expertise and cost. For example, Bob, $John$, and Ana are software analyst. After performing the individual assessment, their evaluations are $Bob = 0.549$, $John = 0.525$, and $Ana = 0.514$.

Group Assessment: To determine the group evaluation of the resources that belong to a determined team, we consider the same metrics that are used in the individual evaluation, but restricting the information to those process instances in which the same team participated previously. To that end, we use the resource process cube $\mathbb{Q}(team\_filter(L, \mathbf{tc}, \mathbf{tr}))$, which only contains information related to process instances of similar characteristics in which the team worked together in order to execute them. Similar to the individual case, a group of lists for each metric is created, each list containing the value of each resource in the team, and weighting each metric by previously specified weights. For example, in the software development process, $Bob(analyst)$, $Teo(designer)$, $Dave(developer)$, and $Alice(tester)$ are the best resources evaluated individually in each life cycle phase, respectively. However, if we consider the collective performance when the resources have worked together as a team in the past, the best evaluated resources would be: *John*, *Pete*, *Sean*, and *Sara*.

Team Recommendation: We propose a recommender system that uses a method based on BPA2 to find the most relevant k-items of a multi-dimensional dataset, i.e., to obtain a ranking of the most suitable teams to be recommended. BPA2 implicitly calculates the overall score for each data item, maintaining in a temporal set the k-data items with the highest overall score. The algorithm allows an iterative approach to access and evaluate the resources based on their local score and their position in each list. If at the same point the temporal set contains k-data items whose overall scores are higher than or equal to a generated threshold, then there is no need to continue scanning the rest of the lists. With BPA2, no position in a list is accessed by the algorithm more than once, which prevents re-accessing data items considering sorted or random access. The output of the algorithm is an ordered list, where the final score for each resource is stored. The first value represents the tuple of resources with the highest overall score and therefore the best recommendation. More details about BPA2 can be found in [2].

Definition 4 ***(Team Resource Recommendation).*** *Let* $\mathbf{tc}$ *be a tuple of* n *resource request characterizations that must be performed and the Cartesian product* $T^{TR} = R^1 \times \cdots \times R^n$ *be the set of all* n*-tuples of resources that can be allocated to the requests in* $\mathbf{tc}$*. A team resource recommendation will propose a* $top - k$ *set of tuples of resources* $TR' \subseteq T^{TR}$*, such that* $|TR'| = k$ *and* $\forall \boldsymbol{tr'} \in TR'$ *and* $\forall \boldsymbol{tr''} \in T^{TR} \setminus TR'$ $score(\boldsymbol{tr'}, \boldsymbol{tc}) >= score(\boldsymbol{tr''}, \boldsymbol{tc})$*. Each tuple* $\boldsymbol{tr} \in TR'$ *represents the allocation of a resource* $tr_i \in \boldsymbol{tr}$ *to a characterization* $tc_i \in \boldsymbol{tc}$*,* $\forall i = 1, \ldots, n$*.*

A method based on BPA2 is used to solve this problem. Here, the elements of the lists are defined for each tuple $\mathbf{tr} \in T^{TR}$. The lists considered are:

- Lists that measure the individual behavior of the resources: $IL_{1,1}, \ldots, IL_{1,m}, \ldots, IL_{i,j}, \ldots, IL_{n,m}$, being $i = 1, \ldots, n$ the different resource characteristics considered, and $j = 1, \ldots, m$ the different metrics considered. $IL_{i,j}$ contains $|R^1| \cdot \cdots \cdot |R^n|$ pairs of the form $(\mathbf{tr}, metric_j(\mathbb{Q}(L), tr_i, tc_i) \cdot w_j \cdot ind\_w_j)$ where $\mathbf{tr} \in T^{TR}$, and w_j is the weight given to the metric j. $IL_{i,j}$ is sorted in descending order. The value of the metrics are computed based on the resource process cube $\mathbb{Q}(L)$.
- Lists that measure the behavior of the resources on a specific team: $GL_{1,1}, \ldots, GL_{1,m}, \ldots, GL_{i,j}, \ldots, GL_{n,m}$, being $i = 1, \ldots, n$ the different resource characteristics considered, and $j = 1, \ldots, m$ the different metrics considered. $GL_{i,j}$ contains $|R^1| \cdot \cdots \cdot |R^n|$ pairs of the form:
  $(\mathbf{tr}, metric_j(\mathbb{Q}(team\_filter(L, \mathbf{tr}, \mathbf{tc})), tr_i, tc_i) \cdot w_j \cdot grp\_w_j)$, where $\mathbf{tr} \in T^{TR}$, and w_j is the weight given to the metric j. $GL_{i,j}$ is sorted in descending order. The value of the metrics are computed based on the resource process

cube $\mathbb{Q}(team\_filter(L, \mathbf{tr}, \mathbf{tc}))$, i.e., the resource process cube that represents the information about those cases where the team **tr** has worked together performing the different occurrences of the resource request characterization **tc** associated to the process registered in the event log L.

Notice that the length of all lists is $|R^1| \cdot \cdots \cdot |R^n|$. This is a requirement for using BPA2. The specified weight for each metric w_j is decomposed so that part of the weight is related to the respective $IL_{*,j}$ list and the other part is related to the corresponding $GL_{*,j}$ list, such that $ind\_w_j + grp\_w_j = 100\%$. The overall score $score(\mathbf{tr}, \mathbf{tc})$ is the weighted value of the different m-metrics obtained by the n-resources tuple **tr** when assigned to the characterization **tc** (2):

$$\begin{aligned} score(\mathbf{tr}, \mathbf{tc}) = & \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} metric_j(\mathbb{Q}(L), tr_i, tc_i) \cdot w_j \cdot ind\_w_j \\ & + \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} metric_j(\mathbb{Q}(team\_filter(L, \mathbf{tc}, \mathbf{tr})), tr_i, tc_i) \cdot w_j \cdot grp\_w_j \end{aligned} \tag{2}$$

4 Empirical Evaluation

We adapted our evaluation considering the Waterfall life cycle as a team allocation scenario. Below, we present the dataset and the configuration of the experiments, and the discussion of the obtained results.

Datasets and Experimental Setup: For our experiments, we considered a synthetic event log with 1,000 cases. We also considered the following attributes related to the software development process: Case ID, Role, Typology, Resource, Overall quality, Incurred cost by phase, Phase creation date, Phase closing date. We created the matrices that describe the expertise required to execute each phase and the level of expertise of each resource, and an array that describes the effective availability of each resource at the moment of making the recommendation. An extension to the OpenXES library is used to standardize and represent the historical and contextual information [1]. We used 20 resources, who perform the role of analysts, designers, developers, and testers. We reproduced three different scenarios in which it was necessary to perform a team recommendation for projects developed according to the Waterfall model. For the resource request characterization, we considered as allocation unit the first four phases of the life cycle. We focused on a single project typology: mobile application development. Our evaluation involves the following experiments:

(a) Calculate the top-3 teams. Weights were specified so as to give more importance to the individual behavior of the resources through the different phases.
(b) We used a similar scenario to the one described in experiment (a), but giving more importance to the group behavior of the resources when they work together through the project phases.

[1] https://svn.win.tue.nl/trac/prom/browser/Packages/ResourceRecommendation/Trunk/src/org/processmining/resourcerecommendation/utils/resrecxes.

(c) Personalized scenario were created, where weights were specified in accordance to the project being developed.

Discussion: To perform the recommendation, we evaluate the resources based on the defined metrics. We specified different weights w_j in accordance to the level of importance given to each $metric_j$. The weight is decomposed, so that a part of the weight is related to each individual assessment list $IL_{*,j}$ and the other part to each group assessment list $GL_{*,j}$. These weights are applied to the metric score for each list of each metric. We used the BPA2 algorithm to find the resource tuples that have the highest overall score. BPA2 produces an ordered list, wherein the final score for each tuple of resources is stored. The top-k tuples represent the most suitable teams to be recommended. For the three experiments, we specified the following weights for the metrics: performance: 20 - quality: 50 - cost: 20 - other dimensions: 10. Particularly, an individual weight ind_w_j of 90% and a group weight grp_w_j of 10% were specified in the experiment (a); 10% individual and 90% group weights for the experiment (b); and 80% individual and 20% group weights for the experiment (c). When creating the event log, we simulated the existence of one resource whose individual evaluation was better when compared to the other resources that participated in the same phase, i.e., there is a resource that is individually better executing each one of the phases. In a similar manner, the existence of teams that are better working together in comparison to other teams was simulated.

Table 1. Experiments results

Exp.	Recommended Team	Phase 1 Analysis	Phase 2 Design	Phase 3 Coding	Phase 4 Testing	Overall score
(a)	Top-**1**:(R1,R5,R9,R12)	0.549	0.539	0.522	0.541	**0.538**
	Top-**2**:(R1,R4,R9,R12)	0.549	0.535	0.522	0.541	**0.537**
	Top-**3**:(R1,R5,R9,R10)	0.549	0.539	0.522	0.536	**0.536**
(b)	Top-**1**:(R2,R4,R8,R12)	0.525	0.535	0.512	0.541	**0.610**
	Top-**2**:(R1,R4,R9,R10)	0.549	0.535	0.522	0.536	**0.590**
	Top-**3**:(R1,R6,R7,R10)	0.549	0.509	0.511	0.536	**0.573**
(c)	Top-**1**:(R1,R4,R9,R10)	0.525	0.535	0.522	0.536	**0.547**
	Top-**2**:(R1,R5,R9,R12)	0.549	0.539	0.522	0.541	**0.544**
	Top-**3**:(R1,R5,R8,R10)	0.549	0.539	0.512	0.536	**0.538**

(*) Phase scores correspond to the individual score of the resources allocated to each phase.

Table 1 shows the results obtained in the experiments. Experiment (a) shows that the resources defined as the best individually are effectively the recommended team (top-1) according to the ranking. In this particular case, the most suitable resources are: the analyst R1, the designer R5, the developer R9, and the tester R12. For experiment (b), we focused on forming the teams considering how well the resources had worked collectively in the past. The results are the expected ones, according to the construction of the experiments. After performing the experiment, the results recommended as the most suitable team the resources: R2, R4, R8 and R12, which represents the group of resources with the best performance working together in comparison to the rest of the teams. In experiment (c) we adjusted the given weights in accordance to specific priorities. The results obtained are different to those generated in experiment (a) and (b), which proves that our approach produces the recommendation based on the characterization of each request, generating diverse results that adapt to specific business contexts. R1, R4, R9 and R10 form the team that best adjusts to the specified priorities. Note that in each experiment a ranking is recommended with the resource tuples to be allocated in accordance to the top-k queries defined. Thus, the person in charge of making the allocation has prioritized team alternatives to allocate, with the possibility of including, if necessary, other subjective criteria not considered as part of the framework. The applicability of our approach can be supported through today corporate information systems (e.g., ERP SAP), which allow the storage and extraction of historical contextual information as well. For usual scenarios where the overall process can be decomposed into disjoint subsets (e.g., activities, sub-process, or phases) our approach is able to generate a ranking of feasible work teams. For more adaptive and complicated scenarios (e.g., agile software development), which might involve a overlapping, incremental, or iterative phases, our framework needs to be adapted.

5 Conclusions and Future Work

We extended our Framework for Recommending Resource Allocation based on Process Mining. The main contributions of this work are as follows. First, we extended our previous framework from recommending single resource rankings to recommending teams that work better collaboratively. Second, our framework is focused on improving decision making, helping the person in charge of forming teams to optimize the use of available human resources. Third, we based the team recommendation on the evaluation of multiple criteria that measure the past behavior of the resources, their expertise and current workload. Moreover, our strategy to evaluate teams combines the evaluation of the individual behavior of the resources and their behavior when they collaborate working together as a team. Fourth, due to the flexibility of this approach, the team recommendation can be executed considering any allocation unit, e.g., at the activity level, at the sub-process level, or at the whole process level, allowing it to be adapted and used in different abstraction levels. For usual team recommendation scenarios, the proposed approach generates the recommendation of the most suitable work teams based on the definition of a resource allocation request at run-time, and the use of contextual and historical information. However, for more complex scenarios, our framework needs to be adapted in order to allocate teams dynamically. Future work consider the creation of heuristics that enable determining faster those teams more likely to be recommended in accordance to the given characterizations, in order to optimize the process for computing the results. Also, we will examine other criteria to measure collaborative work which could be considered as part of our approach, as well as integrate information about the responsibility degree for each activity [7]. We have evaluated our approach over a proof-of-concept implementation. We aim to evaluate the effectiveness and efficiency of our technique through case studies using real-life event logs.

Acknowledgments. This project was partially funded by the Ph.D. Scholarship Program of CONICYT Chile (Doctorado Nacional/2014-63140181), Universidad de Costa Rica and by Fondecyt (Chile) Project No.1150365.

References

1. van der Aalst, W.M.P., Verbeek, H.M.W.: Process discovery and conformance checking using passages. Fundam. Inform. **131**(1), 103–138 (2014)
2. Akbarinia, R., Pacitti, E., Valduriez, P.: Best position algorithms for efficient top-k query processing. Inf. Syst. **36**(6), 973–989 (2011)
3. Arias, M., Rojas, E., Munoz-Gama, J., Sepúlveda, M.: A framework for recommending resource allocation based on process mining. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015. LNBIP, vol. 256, pp. 458–470. Springer, Cham (2016). doi:10.1007/978-3-319-42887-1_37
4. Ballesteros-Pérez, P., González-Cruz, M.C., Fernández-Diego, M.: Human resource allocation management in multiple projects using sociometric techniques. Intl. J. Project Manage. **30**(8), 901–913 (2012)
5. Barreto, A., de Oliveira Barros, M., Werner, C.M.L.: Staffing a software project a constraint satisfaction and optimization-based approach. Comput. OR **35**(10), 3073–3089 (2008)
6. Britto, R., de Alcântara dos Santos Neto, P., Rabelo, R.A.L., Ayala, W., Soares, T.: A hybrid approach to solve the agile team allocation problem. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 1–8 (2012)
7. Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: Automated resource assignment in BPMN models using RACI matrices. In: Meersman, R., et al. (eds.) OTM 2012. LNCS, vol. 7565, pp. 56–73. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33606-5_5
8. Cabanillas, C., Resinas, M., Mendling, J., Cortés, A.R.: Automated team selection and compliance checking in business processes. In: Proceedings of the 2015 International Conference on Software and System Process, ICSSP, pp. 42–51 (2015)
9. Chaudhuri, S., Dayal, U.: An overview of data warehousing and olap technology. ACM Sigmod Rec. **26**(1), 65–74 (1997)
10. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Heidelberg (2013)
11. Gerogiannis, V.C., Rapti, E., Karageorgos, A., Fitsilis, P.: Human resource assessment in software development projects using fuzzy linguistic 2-tuples. In: Artificial Intelligence, Modelling and Simulation (AIMS), pp. 217–222. IEEE (2014)
12. Huang, Z., van der Aalst, W.M.P., Lu, X., Duan, H.: Reinforcement learning based resource allocation in business process management. DKE **70**(1), 127–145 (2011)
13. Huang, Z., Lu, X., Duan, H.: Mining association rules to support resource allocation in business process management. Expert Syst. Appl. **38**(8), 9483–9490 (2011)

14. Kim, A., Obregon, J., Jung, J.-Y.: Constructing decision trees from process logs for performer recommendation. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 224–236. Springer, Cham (2014). doi:10.1007/978-3-319-06257-0_18
15. Kumar, A., Dijkman, R., Song, M.: Optimal resource assignment in workflows for maximizing cooperation. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 235–250. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40176-3_20
16. Li, C., Akker, J.M., Brinkkemper, S., Diepen, G.: Integrated requirement selection and scheduling for the release planning of a software product. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 93–108. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73031-6_7
17. Liu, X., Chen, J., Ji, Y., Yu, Y.: Q-learning algorithm for task allocation based on social relation. In: Cao, J., Wen, L., Liu, X. (eds.) PAS 2014. CCIS, vol. 495, pp. 49–58. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46170-9_5
18. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Single-entry single-exit decomposed conformance checking. Inf. Syst. **46**, 102–122 (2014)
19. Narendra, N.C., Ponnalagu, K., Zhou, N., Gifford, W.M.: Towards a formal model for optimal task-site allocation and effort estimation in global software development. In: 2012 Annual SRII Global Conference, pp. 470–477 (2012)
20. Oberweis, A., Schuster, T.: A meta-model based approach to the description of resources and skills. In: AMCIS, p. 383 (2010)
21. Royce, W.W.: Managing the development of large software systems. In: proceedings of IEEE WESCON, vol. 26, pp. 1–9 (1970)
22. Russell, N., Aalst, W.M.P., Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: identification, representation and tool support. In: Pastor, O., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005). doi:10.1007/11431855_16
23. Schönig, S., Cabanillas, C., Jablonski, S., Mendling, J.: A framework for efficiently mining the organisational perspective of business processes. DSSs **89**, 87–97 (2016)
24. e Silva, L., Costa, A.P.: Decision model for allocating human resources in information system projects. Intl. J. Proj. Manage. **31**(1), 100–108 (2013)
25. Sommerville, I.: Software Engineering. Pearson, London (2015)
26. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: Portfolio-based algorithm selection for SAT. CoRR abs/1111.2249 (2011)
27. Zhao, W., Zhao, X.: Process mining from the organizational perspective. In: Wen, Z., Li, T. (eds.) Foundations of Intelligent Systems. AISC, vol. 277, pp. 701–708. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54924-3_66