Information Systems Frontiers (2021) 23:753–772
https://doi.org/10.1007/s10796-020-09990-7

# OrgMiner: A Framework for Discovering User-Related Process Intelligence from Event Logs

Amit V. Deokar[1] · Jie Tao[2]

Published online: 8 April 2020

**Abstract**
Process Intelligence refers to the extraction and analysis of valuable knowledge nuggets embedded in business process instances/ event logs or enterprise applications, for the purpose of supporting various decision-making processes. Researchers and practitioners mine such event logs using Process Mining and Analytics (PMA) techniques that help analyze business processes across three perspectives: control flow, organization, and data. While previous PMA studies have made advances toward the control flow and data flow perspectives, there is limited research toward the organizational perspective of process intelligence. In this study, we propose an organizational mining framework, *OrgMiner*, that supports constructing organizational models from event logs. The framework utilizes the notion of behavioral patterns, which rely on the weak order relations appearing in event logs. The various modules and knowledge elements in the framework are described in detail. The components of the framework support identifying, selecting, and applying behavioral patterns using different metrics for organizational mining purposes. The derived organizational models can be used to support decision making in scenarios such as task assignment, resource allocation, as well as role-based access control. Compared to extant studies, the proposed approach does not assume prior availability of explicit process models. Additionally, the process patterns presented in this study can be used as building blocks, so that researchers and practitioners can use them directly or extend them further to identify complex organizational processes. A case study is presented to evaluate the feasibility and effectiveness of the *OrgMiner* framework.

**Keywords** Business process management · Process analytics · Behavioral patterns · Process mining · Organizational mining

## 1 Introduction

Business processes present a wealth of data with a potential to be derived into information, and ultimately knowledge for the purposes of management. Extracting and understanding process related knowledge is one of the key challenges in Business Process Management (BPM) domain. In that regard, organizations employ a variety of techniques to improve decision-making that ultimately relies on business process data and observations. These process observations also termed as event logs, come from various information systems supporting the corresponding business processes, sometimes in the form of full-fledged enterprise systems or as standalone applications/systems that capture execution information related to certain process segments or tasks (Ghattas et al. 2014). Process Mining and Analytics (PMA) techniques facilitate *discovery, monitoring*, and *improvement* of existing processes based on the analysis of event logs (van der Aalst 2012a, b). PMA techniques can be applied to various phases in the business process lifecycle to facilitate organizational learning based on process-related knowledge. PMA has been recognized to augment current Business Intelligence (BI) techniques by investigating activities consisting the business processes rather than treating them as *black boxes*, and thus addressing a limitation of mainstream BI applications of mainly focusing on aggregating data for assisting high-level tactical or strategic decision making (Bucher et al. 2009). Thus, with the assistance of PMA techniques, we can effectively transition from BI to Process Intelligence (PI) (van der Aalst et al. 2015).

✉ Amit V. Deokar
amit_deokar@uml.edu

Jie Tao
jtao@fairfield.edu

1 Department of Operations & Information Systems, Manning School of Business, University of Massachusetts Lowell, Lowell, MA, USA

2 Department of Information Systems & Operations Management, Dolan School of Business, Fairfield University, Fairfield, CT, USA

While researchers have primarily focused on applying PMA techniques for studying control-flow and data-flow perspectives of business processes, there is limited work on the organizational perspective (van der Aalst 2011). Organizational and social structures underlying business processes shed light on the knowledge transferring network of task handlers, and as such need to be well-understood for improving the performance of business processes. This information can be valuable in discovering missing knowledge elements or redundant tasks in business processes (Leyer et al. 2016). Existing research relies on explicit process models and/or organizational models for obtaining information on the organizational setting and interactions among knowledge workers (Song and van der Aalst 2008; Thomas and Fellmann 2006). However, in many cases (such as inter-organizational workflows and cross-department team-based processes), neither are process/organizational models explicitly available nor does a formalized structure exist to represent such knowledge. Thus, an alternative approach is deemed necessary in order to discover user-related process intelligence from event logs.

Consider a following real-world scenario that can benefit from the identifications of process patterns. In an organization (e.g. a financial institute), manager A always reviews the transactions completed by trader B and returns tasks back to B, but never reviews the transactions completed by trader C. Such interleaving (i.e. between A and B) and exclusiveness (i.e. between A/B and C) pattern strongly suggests that A and B should be assigned to the same organizational unit (e.g., team), while A/B and C would rather be in different organizational units. Further, if A and B always conduct consecutive tasks together, then the same resource (e.g. a trading terminal) can be allocated to both of them since such allocation will not affect the efficiency of the process. This example gives a preview of basic types of process patterns and their applications to formulate role assignment and resource allocation rules, which are key tasks in organization mining, or other integrations of the business rules (Kluza and Nalepa 2018).

This study focuses on organizational mining, which entails extracting information on organizational structures/models that provide insights on how knowledge workers interact to execute business processes. Organizational mining contributes to important facets of business processes, including *strategy*, *governance*, *people*, and (organizational) *culture*. Moreover, organizational mining can help researchers and practitioners construct ontologies of various industries that represent containers of domain knowledge. In particular, this study contributes to the PMA body of knowledge by presenting: a) a pattern recognition and matching approach in the form of *Behavioral Pattern Discovery Algorithm* (*BPDA*) for identifying behavioral process patterns from event logs, b) a novel organizational mining approach consisting of the *Org-AHC* algorithm based on behavioral process patterns, and c) the *OrgMiner* framework that encapsulates three modules, namely *pattern definition*, *pattern selection*, and *organizational mining* in a systematic manner through links to various pertinent knowledge elements. These design artifacts have been validated through a case study consisting of a large event log, which also demonstrates their feasibility and utility. It is also observed in the case study that our approach is more economically feasible in resource-limited scenarios.

The structure of this paper is organized as follows. Section 2 provides background and overview of PMA, with emphasis on the organizational perspective. A motivating and running example used throughout the paper is discussed in Section 3. Key design artifacts including the *OrgMiner* framework, behavioral pattern recognition using the BPDA algorithm, and organizational mining approach using the *Org-AHC* algorithm, are presented in Section 4. We validate the design artifacts through a case study in Section 5. Discussion on related work in order to situate the key contributions of this study is presented in Section 6. The article concludes with summary remarks in Section 7.

## 2 Related Studies

### 2.1 Process Mining and Analytics (PMA)

PMA techniques have been used to extract embedded process knowledge and to assess and improve the performance of business processes from different perspectives (Caron et al. 2013). The essence of PMA activities is to investigate event logs with the goal of reasoning about the implicit relationships between process constructs. The ultimate goal of PMA is to derive knowledge nuggets in a user-understandable form that can facilitate predictive process management and decision-making. Given that PMA has its roots in the data mining discipline, its practices are analogous to data mining applications. Notably, data mining techniques derive patterns (as formal abstractions of events in the real world) from datasets (as factual records of observations regarding the events), while PMA techniques analyze process execution data (event logs, as observations of events in certain business processes) to construct the process models (as abstractions of actual business processes) (Tiwari et al. 2008). The key similarity between data mining and PMA is that they both derive *implicit* knowledge from (large) data collections.

Earlier PMA approaches have been based on heuristic algorithms, which identify process patterns based on *intuitive* assumptions and approaches. However, a growing number of PMA methods make use of advanced analytic techniques such as genetic algorithms (van der Aalst et al. 2005), fuzzy logic algorithms (Günther and van der Aalst 2007), temporal analysis based algorithms (Köck and Paramythis 2011), clustering algorithms (Bose and van der Aalst 2010), and pattern recognition based algorithms (Ferreira and Thom 2012). PMA activities can be broadly characterized across two dimensions, namely *functions* and *perspectives* (van der Aalst 2012b). In regard to the functional dimension, PMA techniques entail three major functions: The *discovery* function entails mining the event logs to (*semi-*) *automatically* construct the underlying process models and reason about associated properties (activity originators and data objects), and has been the focus of traditional PMA approaches (van Dongen and van der Aalst 2004). The *conformance* checking function involves deviation detection by comparing the minedprocess model that is constructed from executed process observations with an a priori model. Pattern-based approaches have been used for the conformance checking function of PMA in prior studies (Becker et al. 2016). The *enhancement* function also involves an a priori model; but rather than checking for conformance, it extends the mined process model with a new aspect or functionality (Jareevongpiboon and Janecek 2013). Popular enhancement activities include decision mining, user profiling, and performance analysis, which can be useful for identifying and studying the knowledge transfer patterns among networks of knowledge workers in organizations. Furthermore, recent PMA studies have focused on innovative applications such as collaboration task management within business processes (e.g. attention tracking (Fan et al. 2015). The *perspectives* dimension of PMA activities pertains to inquiring about various aspects of business processes such as the control flow perspective ('how'), the resource flow perspective ('who'), and the data flow perspective ('what'). The *control flow* perspective focuses on the ordering of events in business processes (Bertolini et al. 2011); the *resource* perspective focuses on the originators of the events and how they are related (Song and van der Aalst 2008); while the *data* perspective focuses on the data objects and their values associated with each of the executed events (Sun and Zhao 2013). Process pattern-based approaches have been used in the data flow perspective, such as the use of abstract patterns derived from process instances to assess and control the quality of data (Wahyudi et al. 2018). Similarly, process instances have been mined to identify patterns leading to process nuggets termed as *fragments* that may be conducted either in a distributed or parallel manner (Pourmasoumi et al. 2017)

A majority of extant PMA studies focus on the *control flow* perspective with all three functions; while the classical data mining and BI (e.g. *Online Analytical Processing*, OLAP) areas concentrate on the *data* perspective as applied to the BPM context. Also, prior studies regarding the *resource* perspective of the business processes rely significantly on the availability of explicit knowledge abstractions such as process and/or organizational models (Sellami et al. 2013; Tan et al. 2008). Thus, there is lack of research on techniques for mining organizational information solely from event logs, in absence of such prior available models. In this study, we contribute toward addressing this research gap by proposing a framework aimed at organizational mining objectives in absence of any *explicit* process or organizational structure representations.

## 2.2 Organizational Mining

Organizational mining refers to specific PMA techniques for analyzing *implicit* information about resources (originators, and other entities executing activities) in event logs. Organizational mining can have two key applications: (1) to develop an understanding of the social networks depicting the interactions among different resources in organizational or cross-organizational settings; and (2) to (re)structure cross-functional and/or cross-organizational teams of knowledge workers based on their organizational roles (IEEE Task Force on Process Mining 2011). For instance, Alirezaei and Parsa (2018) use Event-Condition-Action (ECA) rules to model cross-organizational, unstructured business processes and the collaborative network within them. Similar to traditional PMA, organizational mining also entails functions including *discovery*, *conformance checking*, and *enhancement*. *Discovery* in organizational mining refers to construction of *organizational models* or *social network models* that reflect the organizational settings from the event logs. Organizational models usually contain information regarding *organizational units* (e.g., departments), *roles* (e.g., tasks), *originators* (e.g., employees), and *relationships* (e.g., A is "part-of" B, or C "is-a" D), while social network models depict the interactions (e.g., handover of work, etc.) among different originators. Also, organizational mining can be used in tandem with other related information to discover *job assignment rules, resource allocation rules*, or *user profiling rules*, for decision support purposes. Within such context, *conformance checking* refers to comparing discovered models/rules with corresponding heuristic counterparts. Moreover, enhancement in the context of organizational mining refers to enriching existing models/rules with value-adding knowledge, such as constructing abstractions (i.e. models, rules) based on different organizational units/roles.

Researchers in the PMA domain have recognized the importance of organizational modeling and have proposed several approaches. For instance, Sellami et al. (2013) suggest the use of an *organizational ontology* for semantically annotating event logs to discover relationships among originators of activities. While this is a valid approach, it is limited by its reliance on *a static underlying knowledge structure*, e.g. the organizational ontology. Song and van der Aalst *(*2008*)* propose an approach for *conformance checking* of organizational models, which relies on *existing explicit process* and *organizational models*. The relatedness between different originators of activities is calculated based on the frequencies of different originators executing similar actions, eventually recommending a grouping of such originators in the same organizational unit. In this study, we take a discovery-oriented approach of organizational mining, in which we do not assume that the existence of *explicit models* (either *process models* or *organizational models*), given that such models may be inaccessible in practice (e.g., cross-organizational processes). Further, in our proposed approach, in addition to task execution frequencies of different originators, we also rely on *recurring behavioral patterns* from process segments recorded in the event logs to draw inferences regarding associations among originators, and in turn about an underlying organizational model. In using process patterns within our approach, we build on the extant work in the workflow patterns area including action patterns notions used for process mining proposed by Smirnov et al. (2012).

## 3 A Running Example

To illustrate the proposed framework in a self-contained manner, we use an event log of the "reviewing process" documented by van der Aalst (2011). The example describes the reviewing process for an academic journal. When a paper is submitted to the journal, it is sent to three different reviewers ('invite reviewers', Activity A). Each of the reviewers is responsible to critique a paper ('get review 1-3', Activity B1–3). After the comments are submitted ('collect reviews', Activity C), a person is responsible to read the reviewers' comments and make a decision ('decide', Activity D). However, sometimes reviewers do not respond to the reviewing requests. In those cases, additional reviewers are invited ('invite additional reviewer', Activity E). The additional reviewer critiques the paper ('get review X', Activity F) and then the comments are analyzed to infer a decision ('decision', Activity D). These steps (E-F-D) are repeated until enough reviews are solicited. Then a final decision is made ('accept', Activity G or 'reject', Activity H).

The original event log contains 10,000 instances (manuscript submissions) and 236,360 events (in aforementioned 8 distinct event classes A through H). For the running example, we select a small sample of this event log, referred to later as the *example log*. Table 1 shows the example log consisting of randomly selected six instances (each row corresponds to a different instance identified by the *instance ID*) that contain all 8 distinct event classes.

We denote each event by two elements: the activity (e.g. 'C') and its originator (e.g. 'Anne'). In total, 10 distinct originators are involved in these six instances. The events in each instance are ordered according to the temporal sequence reflected in their timestamps. The event log is pre-processed in two steps. First, since the *status changes* of the events are less relevant for organizational mining objectives, we choose only the '*completed'* events and ignore events with other statuses. Next, we filter out the events without originators (i.e., the originators of all the *'time-out'* events are tagged as *'_INVALID_'* and all such events are excluded from the example). For illustration purposes, we assume that the six instances exhaustively represent the process, i.e., no other instance exists. We also assume that the underlying process model and the organizational model are unknown/implicit. Based on the example process log, the final organizational model obtained through organizational mining is shown in Fig. 1. In the remaining sections, we describe key aspects of the proposed organizational mining framework using the example log, demonstrate intermediate results obtained while generating the final organizational model, and the interpretation of the organizational model.

## 4 Design and Development of the *OrgMiner* Framework

We first formally define the notion of event logs.

**Definition 1: (Event Logs).** Let $\mathcal{A}$ ($\mathcal{A} = \{\dashv_1, \ldots, \dashv_n\}, n = |\mathcal{A}|$ ) be the set of distinct activities in a business process, and $\mathcal{O}$ ($\mathcal{O} = \{o_1, \ldots, o_m\}, m = |\mathcal{O}|$ ) be the set of all distinct originators (e.g. persons, system modules) that conduct the activities in $\mathcal{A}$. The Cartesian product $\mathcal{E} = \mathcal{A} \times \mathcal{O}$ represents a set of events indicating association of activities and originators (e.g. (*D*, *Wil*) implies that activity *D* is conducted by originator *Wil*). $\mathcal{E}^+$ is the set of possible non-empty finite ordered sequence of events from $\mathcal{E}$. Any $t \in \mathcal{E}^+$ is a possible trace (temporally ordered sequence of events). Thus, an *event log* $\mathcal{L}$ is a multi-set of traces. Each $l \in \mathcal{L}$ is a process instance (a one-time execution of the process). We define the following related notions:

- $\mathcal{R} = [r_{i,j}]$ *is a symmetric, non-empty matrix that reflects the relations between each pair of activities in* $\mathcal{A}$*;* i.e. $r_{i,j} = (a_i, a_j), \forall a_i, a_j \in \mathcal{A}$ *;*
- $\pi_A$ *is an assignment operation applicable to an originator that returns all the activities executed by that originator,* e.g. $\pi_A(o_i) = \{a_i \mid a_i \in \mathcal{A}, i = p, \ldots, q\}$ *;*
- *Given an event* $e = (a, o) \in \mathcal{E}$*,* $\pi_E$ *is an operation that retrieves the activity that was completed during that event,* e.g. $\pi_E(e) = a \in \mathcal{A}$ .

The following assumptions are made regarding the event logs:

- *Each of the events in an event log has an originator; events without any originator (typically automated tasks) are excluded.*

**Table 1** Event log for the running example (Example log)

| Instance ID | Activities and Originators |
|---|---|
| 1 | (A, Mike), (B2, Carol), (B3, Pam), (B1, John), (C, Anne), (D, Wil), (E, Anne), (F, Pam), (D, Wil), (E, Mike), (F, Mary), (D, Wil), (E, Mike), (F, Sara), (D, Wil), (H, Anne) |
| 10 | (A, Anne), (B2, Sara), (C, Mike), (D, Wil), (E, Mike), (D, Wil), (E, Mike), (D, Wil), (E, Anne), (F, Pete), (D, Wil), (E, Anne), (D, Wil), (E, Mike), (F, Carol), (D, Wil), (E, Anne), (D, Wil), (G, Mike) |
| 11 | (A, Mike), (B3, Sam), (C, Anne), (D, Wil), (E, Mike), (D, Wil), (E, Mike), (D, Wil), (E, Mike), (D, Wil), (E, Anne), (F, Mary), (D, Wil), (E, Anne), (D, Wil), (E, Anne), (F, Mary), (D, Wil), (E, Mike), (F, Carol), (D, Wil), (H, Anne) |
| 12 | (A, Anne), (B2, Carol), (B3, John), (C, Anne), (D, Wil), (E, Mike), (D, Wil), (E, Anne), (D, Wil), (E, Mike), (F, Pete), (D, Wil), (E, Anne), (F, Pete), (D, Wil), (E, Mike), (D, Wil), (E, Mike), (F, Carol), (D, Wil), (H, Mike) |
| 13 | (A, Anne), (B2, Carol), (B3, John), (C, Anne), (D, Wil), (E, Anne), (D, Wil), (E, Mike), (D, Wil), (E, Mike), (D, Wil), (E, Mike), (F, Carol), (D, Wil), (E, Anne), (D, Wil), (E, Mike), (D, Wil), (E, Mike), (F, Pam), (D, Wil), (E, Anne), (F, Pete), (D, Wil), (E, Mike), (F, Carol), (D, Wil), (H, Mike) |
| 100 | (A, Mike), (B3, Pam), (B1, Sara), (B2, Pete), (C, Anne), (D, Wil), (E, Mike), (D, Wil), (E, Anne), (D, Wil), (E, Anne), (F, Mary), (D, Wil), (E, Mike), (D, Wil), (E, Anne), (F, Mary), (D, Wil), (G, Mike) |

- *Each of the events signifies an activity; events not referring to an activity are discarded as erroneous from the event log(s).*
- *The relatedness between originators is positively correlated with the temporal relationship between activities being executed by them.*

## 4.1 *OrgMiner* Framework

The *OrgMiner* framework is aimed at providing decision support for business process analysts when conducting organizational analysis (e.g., *learning implicit organizational knowledge for process intelligence*). Figure 2 depicts the functional modules within the framework, as well as the knowledge elements used and generated by these modules. *OrgMiner* consists of three modules, namely: *Pattern Definition, Pattern Selection,* and *Organizational Mining.* Additionally, a dashboard acts as an interface between the system and the end users (business process analysts).

Within the *Pattern Definition* module, four basic types of behavioral patterns are articulated as 'pattern schemas': *strict order pattern*, *inversed strict order pattern*, *exclusiveness pattern*, and *interleaving pattern*. Users may create additional *complex* or ad hoc patterns based on them. The event log(s) to be analyzed serves as the input of this module. The *Pattern Selection* module relies on the pattern schemas and the event log to generate a *behavioral relation matrix* using the proposed *Behavioral Pattern Discovery Algorithm (BPDA)*. This matrix essentially captures the instantiation of patterns embedded in the event logs. Selection of patterns is based on *support* and *confidence* as metrics, resulting in (selected) *behavioral patterns*. The *Organizational Mining* module utilizes the discovered behavioral patterns to create *organizational analytical reports*. In this module, *activity distances* and *activity relatedness* are calculated based on the relations between activities and then used as metrics to determine the connection between originators, namely *originator relatedness*. Following this, the module utilizes the proposed *Organizational Mining Algorithm (Org-AHC)* for *discovering* an organizational model by clustering originators with higher

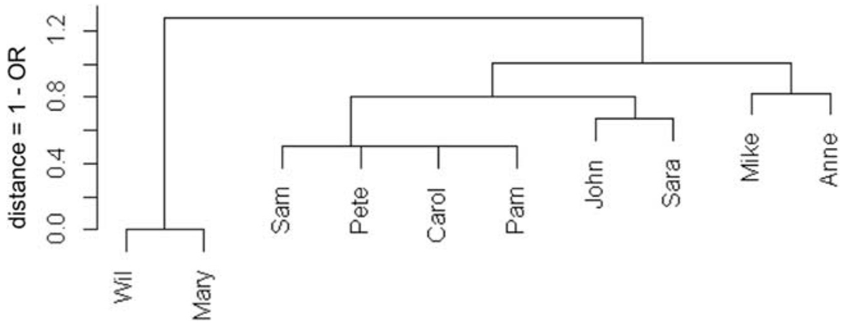


**Fig. 1** Organizational Model Generated for the Example Log

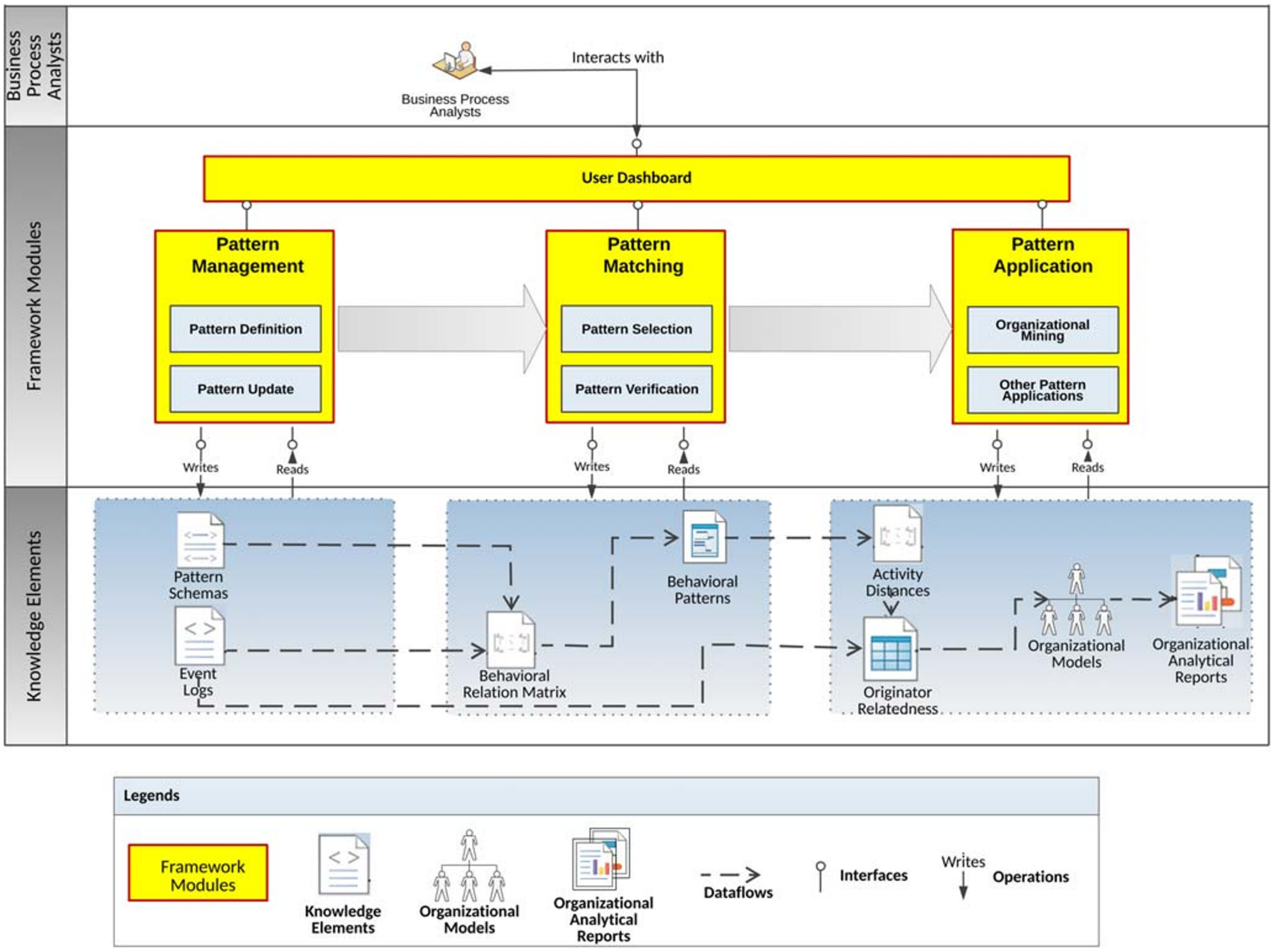


**Fig. 2** *OrgMiner* Framework

*originator relatedness* values together in a bootstrapped manner. Based on the discovered organizational model, *organizational analytical reports* can be generated through (semi-)structured queries for *discovery*, *conformance checking*, and *enhancement* goals. As an example, business process analysts and managers can query about resource allocations to generate an *organizational analytical report* for inferring resource allocation rules. Each of the three modules in the *OrgMiner* framework and their interactions with the knowledge elements are discussed next.

## 4.2 Module I: Pattern Definition

The *Pattern Definition* module concerns tasks such as defining basic behavioral patterns, as well as defining ad hoc or complex patterns based on the basic behavioral patterns. In this article, without loss of generality, we restrict our scope to basic behavioral patterns, while recognizing the extensibility of the approach for complex patterns. Behavioral patterns are based on the concept of behavioral relations, further defined in terms of weak order relations, and analogous to *action patterns* proposed by Smirnov et al. (2012). The major difference is that *action patterns* are defined for business process model repositories, while the proposed behavioral patterns are defined in the context of event logs without any a priori basis of explicit process models.

**Definition 2: (Weak Order Relations).** Let $\mathcal{L}$ be an event log, and $\mathcal{E}^+$ be the set of all possible traces in $\mathcal{L}$. A *weak order relation* $(\succ_{\mathcal{L}}(a_i, a_j))$ contains all pairs of $(a_i, a_j)$ such that $\exists t \in \mathcal{E}^+$, if and only if $\pi_E(e_i) = a_i \wedge \pi_E(e_j) = a_j$, $1 \le i < j \le |t|$, where $t$ is an ordered trace in $\mathcal{L}$. We denote a *weak order relation* as $r_{i,j} = \succ_L^d(a_i, a_j), a_i, a_j \in \mathcal{A}$, which extends the traditional definition by incorporating the notion of distance. The integer $d$ denotes the distance between the activities $a_i$ and $a_j$ in a process instance $l_k$ ($l_k \in \mathcal{L}$), which is measured by the number of activities between $a_i$, $a_j$. Notably, the *weak order relation* is not reversible, i.e. $\succ_L^d(a_i, a_j) \neq \succ_L^d(a_j, a_i)$. Further, $r_{i,j} = \nsucc_{\mathcal{L}}(a_i, a_j)$ denotes that a *weak order relation* does not exist between $a_i$, $a_j$.

We now define three types of behavioral relations depending on how the weak order relation exists in a pair of activities in an event log.

**Definition 3: (Behavioral Relations).** For a process instance $l_k \in \mathcal{L}$ from an event log $\mathcal{L}$, a *behavioral relation* $r_{i,j} \in \mathcal{R}$ has to be one of the following:

- *R1: Strict order relation* ($\longrightarrow_L^d(a_i, a_j)$ ), if and only if $a_i \succ_{\mathcal{L}} a_j \wedge a_j \nsucc_{\mathcal{L}} a_i$;
- *R3: Exclusiveness relation* ($+_L^d(a_i, a_j)$ ), if and only if $a_i \nsucc_{\mathcal{L}} a_j \wedge a_j \nsucc_{\mathcal{L}} a_i$;
- *R4: Interleaving relation* ($\|_L^d(a_i, a_j)$ ), if and only if $a_i \succ_{\mathcal{L}} a_j \wedge a_j \succ_{\mathcal{L}} a_i$.

It can be noted that a strict order relation is not commutative, while the other two behavioral relations exhibit commutative property. Thus, in addition we define an inversed strict order relation as *R2*: $\longrightarrow_L^d(-1)(a_i, a_j)$.

These *behavioral relations* are codified as *pattern schemas* and used in subsequent modules. The use of such behavioral relations can be illustrated with the activities in the example log. For instance, we have $\longrightarrow_L^d(A, D)$ since in all instances Activity A precedes Activity D. Similarly, the following patterns also hold: $\|_L^d(D, E)$ and $+_L^d(G, H)$. Additionally, from instance 1 in Table 1, the following holds: $\longrightarrow_L^3(A, C)$ since there are three activities between Activity A and C. If an activity appears several times in a process instance (e.g. in a loop), $d$ is the longest distance between the two activities. For instance, in instance 10, it exists: $\longrightarrow_L^{16}(A, D)$ since there are maximally 16 activities between A and D.

## 4.3 Module II: Pattern Selection

The second module, *Pattern Selection*, utilizes the pattern schemas from the prior module to identify behavioral patterns from the event log(s) that should be selected for each pair of activity. The module generates a behavioral relation matrix as an intermediate output for further analysis stages including pattern discovery. A *behavioral relation matrix* is defined as follows.

**Definition 4: (Behavioral Relation Matrix).** For an event log $\mathcal{L}$, $R^B = \mathcal{A} \times \mathcal{A}$, $R^B \subseteq \mathcal{R}$ is a non-empty matrix that reflects the maximal behavioral relation and its occurrence between each pair of activities in $\mathcal{A}$; e.g. $r_{i,j}^B = (r_{i,j}, n), \forall r_{i,j}^B \in R^B$, where $r_{i,j}$ is the *behavioral relation* between activities ($a_i$, $a_j$) defined above, while $n$ is the number of its occurrences, i.e., *frequency of a behavioral relation* $r_{i,j}$ in the event log $\mathcal{L}$.

Figure 3(a) illustrates the *Behavioral Pattern Discovery Algorithm* (BPDA) for generating the behavioral relation matrix from an event log considering all pairs of activities across all of its instances. Fig. 3(b) shows a portion of the behavioral relation matrix for the activities $C$, $D$, $E$, and $F$ in the example log, considering the strict order relation. This shows, for instance, that pattern *R1* (strict order relation) exists between activity C and D and occurs 6 times in the example log.

To determine which of the discovered patterns should be selected, *support* and *confidence* metrics are used to calculate *conviction*, the ranking metric for candidate patterns. The pattern selection process is similar in principle to the *Apriori* algorithm for association rules proposed by Savasere et al. (1995) and Agrawal and Srikant (1994), but the definitions of the support and confidence metrics are unique to this application due to the ordering constraints. For deriving the pattern selection threshold, we put higher emphasis on confidence since it indicates the relation that is of greater significance in comparison to others.

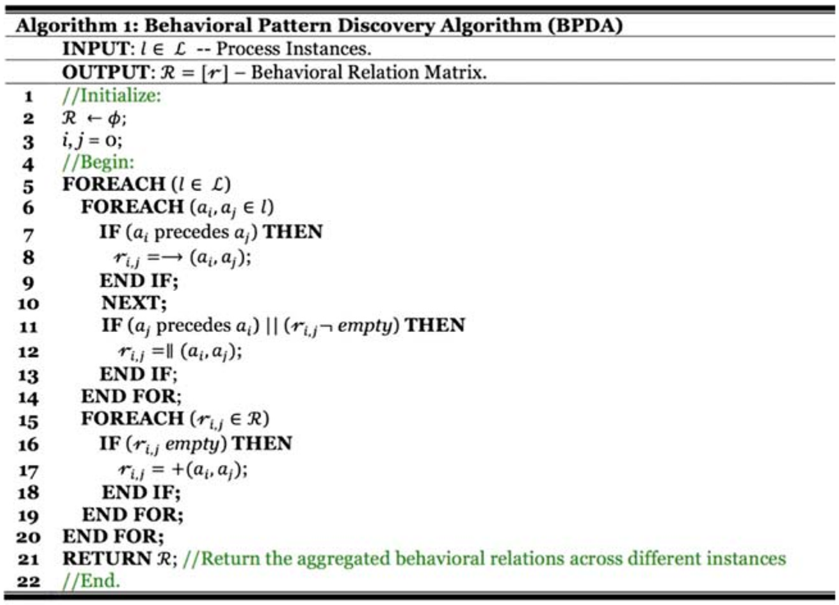
Support of a behavioral pattern $supp(r_{i,j})$ determines how frequently the pattern $r_{i,j}$ appears between activities ($a_i$, $a_j$) across all process instances ($l_k \in \mathcal{L}$ ) in an event log $\mathcal{L}$ (which is $n$ according to Definition 4 above). Confidence of a behavioral pattern is defined as follows. Given $supp(a_i)$ as the support count of activity $a_i$ in event log $\mathcal{L}$ and $\varepsilon$ as a constant, we have:

$$conf(r_{i,j}) = \begin{cases} \dfrac{supp(r_{i,j})}{supp(a_i)}, if \text{ pattern} \in \{R1, R2, R4\} \\ \varepsilon, \quad if\, pattern \in \{R3\} \end{cases}$$

The rationale for splitting the definition of confidence of relationships is twofold. First, the patterns $\longrightarrow_L^d(a_i, a_j)$, $\longrightarrow(-1)(a_i, a_j)$, and $\|_L^d(a_i, a_j)$ align with the notion of association rules, which rely on the *intersection* of the antecedent and the consequent. Thus, we adopt the traditional definition of confidence from association rule learning for relations *R1*, *R2*, and *R4* (Nguyen et al. 2014). Computationally, confidence is the ratio of support of the relation (i.e., $supp(r_{i,j})$ and the support of the preceding activity (i.e., $supp(a_i)$). Second, it is interesting to note that pattern *R3* relies on the *union* of preceding activity ($a_m$) and following activity ($a_n$), which by definition, does not occur. As such, the traditional definition of confidence does not apply directly to pattern *R3*. So, we use a constant ($\varepsilon$) for the confidence of *R3* and the value of $\varepsilon$ is determined experimentally. To illustrate the computations of support and confidence, consider the data in the running example. In order to reason about the relationship *R1* between B3 and B1 ($\rightarrow_L(B3, B1)$) in the event log $L$, it can be noted that this relationship occurs in 2 instances out of the total 6 instances in the example log. We have, $supp$(B3 ) = 5/6, while $supp$(B1 ) = 2/6. Thus, $supp(\rightarrow_L(B3, B1)) = \frac{2}{6} = 0.33$, while $conf(\rightarrow_L(B3, B1)\ ) = \frac{2}{5} = 0.4$.

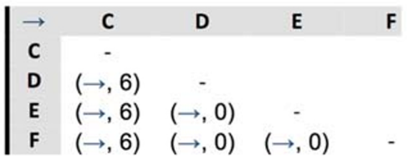
The BPDA algorithm is used to identify the patterns, following which the support and confidence of the patterns between all pairs of activities in $\mathcal{L}$ are computed. Next, if the

**Fig. 3** **a** Behavioral Pattern Discovery Algorithm (BPDA) Algorithm. **b** Example Output (partial) of Application of the BPDA Algorithm



```
Algorithm 1: Behavioral Pattern Discovery Algorithm (BPDA)
    INPUT: $l \in \mathcal{L}$ -- Process Instances.
    OUTPUT: $\mathcal{R} = [r]$ – Behavioral Relation Matrix.
    //Initialize:
    $\mathcal{R} \leftarrow \phi$;
    $i, j = 0$;
    //Begin:
    FOREACH ($l \in \mathcal{L}$)
      FOREACH ($a_i, a_j \in l$)
        IF ($a_i$ precedes $a_j$) THEN
          $r_{i,j} = \rightarrow (a_i, a_j)$;
        END IF;
        NEXT;
        IF ($a_j$ precedes $a_i$) || ($r_{i,j} \neg$ empty) THEN
          $r_{i,j} = \| (a_i, a_j)$;
        END IF;
      END FOR;
      FOREACH ($r_{i,j} \in \mathcal{R}$)
        IF ($r_{i,j}$ empty) THEN
          $r_{i,j} = +(a_i, a_j)$;
        END IF;
      END FOR;
    END FOR;
    RETURN $\mathcal{R}$; //Return the aggregated behavioral relations across different instances
    //End.
```

**(a) Behavioral Pattern Discovery Algorithm (BPDA) Algorithm**



| → | C | D | E | F |
|---|---|---|---|---|
| C | - | | | |
| D | (→, 6) | - | | |
| E | (→, 6) | (→, 0) | - | |
| F | (→, 6) | (→, 0) | (→, 0) | - |

**(b) Example Output (partial) of Application of the BPDA Algorithm**

support and confidence values of the strict order/interleaving patterns are greater than the respective user-defined thresholds, the patterns are accepted, else the exclusiveness pattern is said to hold. We also note here that confidence is later used to calculate activity/originator relatedness (AR/OR), discussed in the next subsection.

In the context of this study, we adopt *conviction* as the metric to rank the learned candidate patterns because it takes into consideration both the support of both the preceding as well as the following activity. Conviction of a pattern $r_{i,j}$ refers to the expected frequency of a pattern between a pair of activities $(a_i, a_j)$:

$$conv\left(r_{i,j}\right) = \begin{cases} \dfrac{1-supp\left(a_j\right)}{1-conf\left(r_{i,j}\right)}, if \text{pattern} \in \{R1, R2, R4\} \\ \dfrac{1}{1-\varepsilon}, \ if pattern \in \{R3\} \end{cases}$$

Here, $supp(a_j)$ is the support of the consequent while $conf\left(r_{i,j}\right)$ is the confidence of the pattern respectively. Conviction is sensitive to the rule direction and is not symmetric like some of other metrics like lift (i.e., $conv\left(r_{i,j}\right) \neq conv\left(r_{j,i}\right)$ ). This is in line the inherent temporal nature between the activities and attempts to measure the degree of implication of the behavioral pattern. Conviction values above 1 mean positive dependence, while values in the (0,1) mean negative dependence. Also, conviction value of 1 implies independence between the antecedent and the consequent, while a theoretical value of infinity indicates logical implication (confidence = 1) (Guillet and Hamilton 2007). Thus, *conviction* is a more appropriate measure to rank candidate patterns. As an illustration, $conv(\rightarrow_L(B3, B1)\ ) = \frac{1-2/6}{1-0.4} \approx 1.1167$, implying a strong positive dependence among the activity pair (B3, B1) for the *R1* relation.

There are several methods to determine the threshold(s) for pattern selection such as: (i) a user-defined value (i.e., threshold, or top-*k* rules based on a specific metric); (ii) a (semi-) automatic procedure to determine the optimal values for the threshold(s). In this study, we use a combination of both approaches to determine the pattern selection thresholds. For the pattern selection purpose, we are interested in activity sets with higher conviction values. Thus, any selected pattern is required to have the highest *conviction* value as compared to other behavioral patterns between the same pair of activities. Thus far, we have discussed domain-independent metrics to evaluate behavioral relations. However, we note that domaindependent metrics may also be derived based on the analysis conducted that is specific to the context at hand as well as by adapting domain-independent metrics. For example, *conviction* also serves as the *Handover-Work ratio* (*HoW ratio*) metric when measuring the robustness of behavioral relations, which would have important practical semantics in different domains (e.g., building knowledge intensive, domain specific semantic repositories (Fraga et al. 2019)). In the healthcare domain, for instance, *HoW ratio* can indicate the likelihood of a patient receiving treatment Y after receiving treatment X in the same visit (process instance), compared to random chance. This can be used for decision-making during resource allocation, or facility/process redesign.

Building on the concept of behavioral relations, we can now define behavioral patterns formally. A tuple $BP = \left(r_{i,j}, d, conv\right)$ denotes a behavioral pattern in an event log $\mathcal{L}$, where:

- $r_{i,j}$ is one of the behavioral relations between ($a_i$, $a_j$) from Definition 3;
- $d$ is the distance between the two activities;
- $conv$ is the conviction $conv\left(r_{i,j}\right)$ of the specific pattern between ($a_i$, $a_j$).

Table 2 illustrates the outcomes of the pattern discovery process for the example log. The matrix is noted to be asymmetric along its diagonal elements. The behavioral relation matrix is obtained using the BPDA algorithm on the example log, and then selecting behavioral patterns with the highest conviction value. The cell for activities $B2$, $H$ list both $R1$ and $R3$ since the conviction values of both the strict order pattern and the exclusiveness pattern are equal. In such cases, business process analysts can either manually assign a behavioral pattern to the pair of activities, or create a new pattern style of interest, in order to resolve such conflicts. In this example, we select the strict order pattern over the exclusiveness pattern between (*B2, H*) for the organizational mining purpose.

**Table 2** Example of selected behavioral patterns

| | Activity 2 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Task | A | B1 | B2 | B3 | C | D | E | F | G | H |
| Activity 1 | | | | | | | | | | |
| A | – | R3 | R2 | R2 | R2 | R2 | R2 | R2 | R3 | R2 |
| B1 | R3 | – | R1 | R3 | R3 | R3 | R3 | R3 | R3 | R3 |
| B2 | R1 | R2 | – | R4 | R2 | R2 | R2 | R2 | R2 | R2/R3 |
| B3 | R1 | R3 | R4 | – | R2 | R2 | R2 | R2 | R3 | R2 |
| C | R1 | R3 | R1 | R1 | – | R2 | R2 | R2 | R3 | R2 |
| D | R1 | R3 | R1 | R1 | R1 | – | R4 | R4 | R3 | R2 |
| E | R1 | R3 | R1 | R1 | R1 | R4 | – | R4 | R3 | R2 |
| F | R1 | R3 | R1 | R1 | R1 | R4 | R4 | – | R3 | R2 |
| G | R3 | R3 | R3 | R3 | R3 | R3 | R3 | R3 | – | R3 |
| H | R1 | R3 | R1/R3 | R1 | R1 | R1 | R1 | R1 | R3 | – |

## 4.4 Module III: Organizational Mining

The behavioral patterns are used as the basis of mining organizational structure from the event logs. From the pattern matching and selection module, we have the behavioral relation matrix $R^B = [r^B]$, which is an $|A| \times |A|$ matrix. To quantify and normalize the relations between each pair of activities ($a_i$, $a_j$), we use the notions of *average behavioral distance* and *activity distance* to arrive at the *activity distance matrix*, discussed next.

Let $T(a_i, a_j)$ be the set of all traces containing activities $a_i$, $a_j$. $\forall t_k(a_i, a_j) \in T(a_i, a_j)$ (k = 1, 2, …, m). If the distance between $a_i$ and $a_j$ is $d_k$, then the *average behavioral distance* ($\overline{d\left(a_i, a_j\right)}$ ) between $a_i$, $a_j$ is given by: $\overline{d\left(a_i, a_j\right)} = \frac{\sum_1^m d_k}{m}$. For example, in example log, $\overline{d(A, C)} = \frac{1+1+1+2+2+3}{6} = \frac{5}{3}$.

The *activity distance* can be computed based on the *average behavioral distance* using the following rationale. If an interleaving pattern holds between ($a_m$, $a_n$), the activity similarity between ($a_m$, $a_n$) would be the highest, so the activity distance between them would be the lowest (for computational purpose, denoted as 0). If an exclusiveness pattern exists between ($a_m$, $a_n$), the activity similarity between ($a_m$, $a_n$) would be the lowest, so the activity distance between them would the highest (for computational purpose, denoted as $+\infty$). If a strict order pattern holds between ($a_m$, $a_n$), then $\mathcal{D}(a_m, a_n) \in (0, +\infty)$. Since $\mathcal{D}(a_m, a_n)$ increases as $\overline{d(a_m, a_n)}$ increases and $conf\left(r_{m,n}\right)$ decreases, we computationally denote $\mathcal{D}(a_m, a_n) = \frac{\overline{d(a_m, a_n)}}{conv\left(r_{m,n}\right)}$. The *activity distance matrix* can be obtained by representing the activity distances in a matrix format.

**Definition 5: (Activity Distance).** The *activity distance* $\mathcal{D}$ is defined as follows:

$$\mathcal{D}(a_m, a_n) = \begin{cases} 0 \ \textit{if} \ \text{R4} \ \textit{exists between} \ (a_m, a_n) \\ \left(\frac{\overline{d(a_m, a_n)}}{conv\left(r_{m,n}\right)}\right) \ \textit{if} \ \text{R1/R2} \ \textit{exists between} \ (a_m, a_n) \\ +\infty \textit{if} \ \text{R3} \ \textit{exists between} \ (a_m, a_n) \end{cases}$$

where *conv* is the conviction of a behavioral pattern between a pair of activities ($a_m$, $a_n$); and $\overline{d}$ is the average behavioral distance between the two activities in the same pattern across different instances. Table 3 shows the *activity distance matrix* between the activities in the example log. The cells highlighted in pale blue reflect the interleaving pattern between the activities (e.g., (B2, B3)). The cells highlighted in dark blue reflect the exclusiveness pattern between the activities (e.g., (A, B1)). The other cells reflect the strict order pattern or

**Table 3** Example of activity distance

| Activity Distance | A | B1 | B2 | B3 | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|---|---|
| **A** | 0.000 | | | | | | | | | |
| **B1** | 0.000 | 0.000 | | | | | | | | |
| **B2** | +∞ | +∞ | 0.000 | | | | | | | |
| **B3** | 0.667 | +∞ | 0.000 | 0.000 | | | | | | |
| **C** | 2.167 | +∞ | +∞ | 0.500 | 0.000 | | | | | |
| **D** | 18.000 | +∞ | +∞ | 16.333 | 14.833 | 0.000 | | | | |
| **E** | 16.167 | +∞ | +∞ | 14.500 | 13.000 | +∞ | 0.000 | | | |
| **F** | 16.167 | +∞ | +∞ | 15.000 | 13.500 | +∞ | +∞ | 0.000 | | |
| **G** | 0.000 | +∞ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | |
| **H** | +∞ | +∞ | +∞ | +∞ | +∞ | +∞ | +∞ | +∞ | 0.000 | 0.000 |

inversed strict order pattern. Building on the concept of activity distance, we characterize the affiliation between activities and originators through the concepts of *activity relatedness* and *originator relatedness* respectively.

**Definition 6: (Activity Relatedness).** *Activity relatedness* (*AR*) is defined as:

$$AR(a_m, a_n) = e^{-\mathcal{D}(a_m,a_n)}$$
$$= \begin{cases} e^0 = 1 & \textit{if R4 exists between } (a_m, a_n) \\ e^{-\frac{d(a_m,a_n)}{conv(r_{m,n})}} & \textit{if } \mathrm{R1/R2} \textit{ exists between activities} \\ e^{-\infty} = 0 & \textit{if } \mathrm{R3} \textit{ exists between activities} \end{cases}$$

Since the activity distances are denoted in the range of [0, +∞], we use the exponential function *e* for normalization such that $AR(a_m, a_n) \in [0, 1]$. Also, *AR* of the same activity is defined to equal unity, i.e., $AR(a_m, a_m) = 1$. Next, in order to map the AR to the relatedness between originators, we apply the assignment operation $\pi_A$. For any pair of originators $o_i, o_j \in \mathcal{O}$, applying the assignment function we have, $\pi_A(o_i) = \{a_i | a_i \in \mathcal{A}, i = p, \dots, q, p < q\}$, $\pi_A(o_j) = \{a_j | a_j \in \mathcal{A}, j = m, \dots, n, m < n\}$. Table 4 shows the *AR* metric between the activities in the example log. It may be noted that the cells highlighted in orange are rounded to 0.

**Definition 7: (Originator Relatedness).** *Originator relatedness* (*OR*) is defined as:

$$OR(o_i, o_j) = \begin{cases} \dfrac{\sum_{i=p}^{q} \sum_{j=m}^{n} AR(a_i, a_j)}{(q-p-1) \times (n-m-1)}, \textit{if } o_i \neq o_j \\ 1, \textit{if } o_i = o_j \end{cases}$$

The *OR* metric is symmetric, i.e., $OR(o_i, o_j) = OR(o_j, o_i)$. Also, in order to achieve more accurate organizational models, two methods can be employed: threshold-based approach or top-*k* approach. A threshold ($\gamma$), for example, on particular activities undertaken by specific originators considers the ratio of occurrence of the activity by the originator to the overall occurrence of the activity. Only *AR* values of the activities greater the threshold are then considered in computing *OR*. The threshold-based approach assumes a strong relationship between the value of the threshold and the quality of the discovered patterns. Also, discovering the optimal value of the threshold is a non-trivial problem. As such, we select the top-*k* method in this study. In particular, *k* refers to the number of rules containing activities undertaken by specific pairs of originators with the highest conviction values. Table 5 shows the *OR* metric between the originators in the example log with the top-3 method.

**Table 4** Example of activity relatedness

| Activity Relatedness | A | B1 | B2 | B3 | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|---|---|
| **A** | 1.000 | | | | | | | | | |
| **B1** | 1.000 | 1.000 | | | | | | | | |
| **B2** | 0.000 | 0.000 | 1.000 | | | | | | | |
| **B3** | 0.513 | 0.000 | 1.000 | 1.000 | | | | | | |
| **C** | 0.114 | 0.000 | 0.000 | 0.607 | 1.000 | | | | | |
| **D** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 1.000 | | | | |
| **E** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 1.000 | | | |
| **F** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 1.000 | | |
| **G** | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | |
| **H** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 1.000 | 1.000 |

The *AD*, *AR*, and *OR* metrics can then be tied to a certain domain. *AD* and *AR* measure how close two activities in a process are, while *OR* measures how close two originators are as they frequently conduct the two activities measured in *AD* and *AR*. Again, considering the healthcare domain as an example, lower *AD* or higher *AR* would reflect that two treatments X and Y often occur sequentially during patient visits; thus, they may be used to measure *treatment consecutiveness*. Higher *OR* values would reflect that two physicians/technicians often conduct treatments *X* and *Y* together, and thus, it may be used to measure the *collaborative consistence* between physicians/technicians. Given the understanding that physicians/technicians with higher *collaborative consistence* should work in the same group or department, various resource-based decisions can be informed.

The aforementioned metrics can lead to a flat organizational model; however, organizational models typically follow a hierarchical fashion. To derive hierarchical organizational models, we apply an *Agglomerative Hierarchical Clustering* (AHC) technique adapted from the group-average AHC algorithm proposed by Shepitsen et al. (2008). Hierarchical clustering has proven to be more efficient than traditional exhaustive search methods (Cong et al. 2015), and is very relevant for organizational mining. The proposed *Org-AHC* clustering algorithm, shown in Fig. 4, conducts stepwise clustering on originators when *OR* between two originators is greater than the threshold (*measure*) at level $k \in [1, |\mathcal{O}|]$. It then moves up to the ($k$ - $1$) level with threshold updated to (*measure* + *step*). The algorithm halts at ($k = 1$), which implies that all originators belong to the same cluster.

Figure 1 depicted earlier, shows the organizational model in the form of a dendogram obtained by applying the *Org-AHC* algorithm for the example log. Given that the example log contains too few (six) instances, the organizational model is clearly not representative to draw general inferences about the business context. Also, some observations are worth noting here. One issue observed in this organizational model is that originator *Mary* should have been grouped with {Pete, Pam} since *Mary* is an external reviewer according to Table 1, while originator *Wil* should have been grouped with {*Anne*, *Mike*} because he is also a decision maker (e.g. *senior editor*) in the review process. The issue can be rationalized as follows. Upon close examination of the data, it is evident that there are some short loops (e.g. *E*-*F*-*D*, *E*-*D*) in the example log. Thus, the pattern between each pair from Activities *D*, *E*, and *F* are all interleaving. Also, in this small example, *Wil* and *Mary* only conducted one activity respectively (‘*Wil*, *D*’, ‘*Mary*, *F*’), which *erroneously* increases their originator relatedness (*OR*). One approach to counteract this issue is to increase the size of instances so that the originators would possibly have more than one activity. Another way to amend this issue is to develop a method to deal with the short loops in event logs, as indicated by Alves de Medeiros et al. (2004). We follow the first approach by selecting the complete “review process” event log (c.f. http://data.3tu.nl/repository/uuid:da6aafef-5a86-4769-acf3-04e8ae5ab4fe).

The complete event log contains 10,000 traces (papers) and 236,360 events (in 8 distinct event classes). 10 originators are involved in the event log. The organizational model in the form of a dendogram obtained by applying the framework for the entire review event log is shown in Fig. 5(a). Using a large event log addresses aforementioned issue of short loops. It can be noted that {*John*, *Sam*, *Carol*, *Sara*} are merged into the same organizational unit (or role) since they undertake similar activities in the example log. Thus, they can be labeled as *in-house reviewers*. Similarly, {*Pete, Mary*, *Pam*} (*external reviewers*) and {*Anne*, *Mike*} (*editors*) can be labeled as such in the organizational model. Fig. 5(b) shows the manually derived organizational model for comparison.

**Table 5** Example of Originator Relatedness

| Originator Relatedness | Anne | Carol | John | Mary | Mike | Pam | Pete | Sam | Sara | Wil |
|---|---|---|---|---|---|---|---|---|---|---|
| Anne | 1.000 | | | | | | | | | |
| Carol | 0.000 | 1.000 | | | | | | | | |
| John | 0.083 | 0.778 | 1.000 | | | | | | | |
| Mary | 0.000 | 0.333 | 0.000 | 1.000 | | | | | | |
| Mike | 0.275 | 0.067 | 0.133 | 0.000 | 1.000 | | | | | |
| Pam | 0.000 | 0.500 | 0.500 | 0.500 | 0.000 | 1.000 | | | | |
| Pete | 0.000 | 0.500 | 0.500 | 0.500 | 1.000 | 0.500 | 1.000 | | | |
| Sam | 0.000 | 0.667 | 1.000 | 0.000 | 0.000 | 0.500 | 0.500 | 1.000 | | |
| Sara | 0.083 | 0.667 | 0.667 | 0.667 | 0.667 | 0.667 | 0.667 | 0.667 | 1.000 | |
| Wil | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 1.000 |

**Fig. 4** *Org-AHC* Algorithm

```
Algorithm 2: The Organizational Mining Algorithm (Org-AHC)
    INPUT: $\mathcal{L}$ -- Event log;
    INPUT: $[OR(o_i, o_j)]$ – The matrix of Originator Relatedness between $o_i, o_j, \forall o_i, o_j \in O$.
    OUTPUT: $Dendrogram_k, k \in [1, |O|]$.
//Initialize:
DEFINE measure; //Define the cluster threshold;
DEFINE step; //The constant subtracted from each step, control granularity;
GET $\mathcal{L}$; //Read event log;
GET $[OR(o_i, o_j)]$;
BEGIN
$C(i) = \{o_i\}, \forall i, o_i \in O$;
FOR ($k = |O|$ ; $k--$; $k > 1$)
  FOREACH ($o_i, o_j \in O$)
    IF ($OR(o_i, o_j) > measure$) THEN
      $Dendrogram_k = \{C(1), \ldots, C(k)\}$;
      $C(i) =$ JOIN $(C(i), C(j)), \forall i, j$;
      DELETE $C(j)$;
    END IF;
  END FOREACH;
  $measure = measure + step$;
END FOR;
DISPLAY $Dendrogram_k$;
END;
```

## 5 Demonstration of *OrgMiner*: Repair Process Case Study

In order to demonstrate the feasibility and efficacy of the proposed approach, we use a case study of repair process of phones in an electronics company. This case study is a real-world dataset used popularly in several process mining research studies, e.g., Bose and van der Aalst (2012).

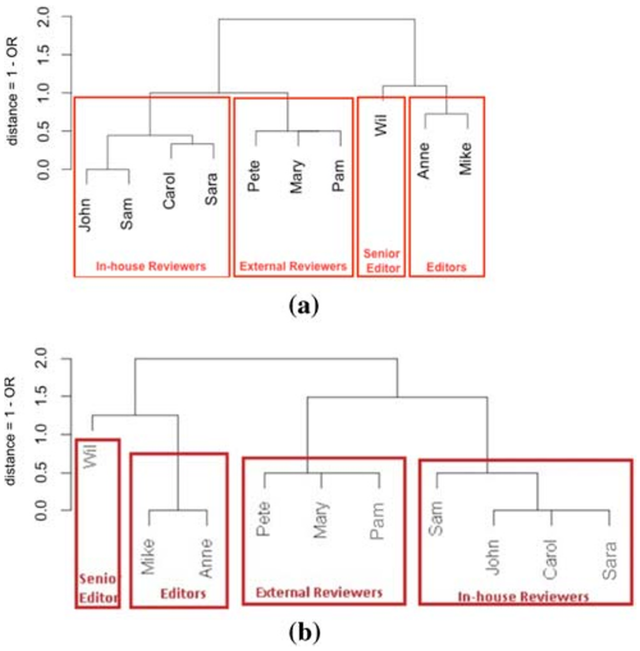


**Fig. 5** **a** Reviewing process: Mined organizational model from entire event log. **b** Reviewing process: Manually derived organizational model

### 5.1 Case Overview

The repair process event log is structurally similar to the review process discussed in the event log. The process model underlying this process is assumed to be unknown or implicit. The process logic in described in this section for readers' understanding and is not incorporated in the demonstrate of the application of the framework. The process typically starts with a repair request registered in the system ('register', Activity A). Then the defected phone is sent for investigation ('analyze defect', Activity B). There are two types of repairs in this process: if the issue with the defected device is minor, then just a simple repair can suffice ('repair (simple)', C1); otherwise, complex repairs are needed ('repair (complex)', C2). Any repair done needs to be tested to ensure that the analyzed defect has actually been fixed ('test repair', D), and one of the two outcomes is expected: successful or failed repair. If the repair is successful, the repair case is archived ('archive repair', E1); if the repair fails, the repair activity (C1 or C2) is done again ('restart repair', E2). These few steps (C1/C2-D-E2) can repeat several times until the repair is tested to be successful. The repair process concludes by informing the customer that their phone has been successfully repaired ('inform user', F).

The repair event log contains 1104 traces (repair instances) and 29,058 events (in 8 distinct event classes). 11 originators are involved in this event log. One of the reasons for selecting this case study is that the roles of all the originators are predefined in the log: the 11 originators are categorized into four roles: {*system*: system, *complex repairers*: solverC1, solverC2, solverC3, *simple repairers*: solverS1, solverS2, solverS3, *testers*: tester1, tester2, tester3, tester4, tester5, tester6}.

## 5.2 Organizational Mining

We conduct organizational mining using the approach discussed in Section 4 on the repair event log. The organizational model generated using the *OrgMiner* framework is shown in Fig. 6(a).

Four organizational groups are evident from the mined organizational model shown in Fig. 6(a). The rightmost group containing {SolverS1, SolverS2, SolverS3} is the 'simple repairers' team. On the left half, three pairs of originators {(Tester1, SolverC1), (Tester2, SolverC2), (Tester3, SolverC3)} are clustered together, and can be labeled as 'complex case handlers'. Such clustering is reasonable since with complex repair claims, frequent/specialized testing is required. The second organizational group, namely 'testers', consists of {Tester4, Tester5, Tester6} and are responsible for testing repairs from both the 'simple repairers' and the 'complex case handlers' teams. Lastly, the 'system' group, contains only one unit, namely {system}, since it represents system automation and is distinct from other originators. The mined organizational model thus reveals implicit knowledge embedded in the event log: the 'complex repairer' group and the 'tester' group are broken down, and some of the testers {Tester1, Tester2, Tester3} are paired with the complex repairers {SolverC1, SolverC2, SolverC3}.

Based on the assumption that no explicit process or organizational models exist for the event log, we conduct an empirical evaluation on the discovered organizational model for its accuracy. Four researchers were tasked with creating an organizational model based on the event log. To keep the evaluation unbiased, the mined organizational model was not revealed to the researchers until they finished creating their own models. The researchers created the originator relatedness matrices which were then reconciled to form the organizational model shown in Fig. 6(b). Next, the *Cophenetic Correlation Coefficient* (CCC) (Fowlkes and Mallows 1983) is used to evaluate the accuracy of the mined organizational model in Fig. 6(a), considering the model in Fig. 6(b) as the ground truth. The CCC values lie in the interval [−1,1], and values close to 1 indicate high resemblance between the two dendograms being compared. The CCC value between the two dendograms in Fig. 6(a) and (b) is 0.766, which provides substantive evidence that the organizational model generated by the *OrgMiner* framework is in accordance with the judgments from domain experts.

## 5.3 Application of Organizational Mining

We next discuss the utility of the *OrgMiner* framework through a popular business process management application of generating resource allocation rules. Resource allocation rules entail mapping activities to appropriate originators. Typically, role-based approaches are employed in resource allocation applications, where originators are divided into high-level organizational groups based on their organizational characteristics (e.g., title) and then assigned to different activities (Liu et al. 2012). Such approaches present the challenge of coarse-grained role definitions which may lead to improper assignments. With the *OrgMiner* framework, we can capture the handover-work relationship between the originators and construct the organizational model (containing the organizational groups) in a fine-grained manner. Moreover, the *OrgMiner* framework also captures the logical association between the workflow activities and the originators, which supports resource allocation decision-making.

In order to use the *OrgMiner* framework for mining resource allocation rules, the event log is analyzed using the BPDA algorithm to generate the behavioral relation

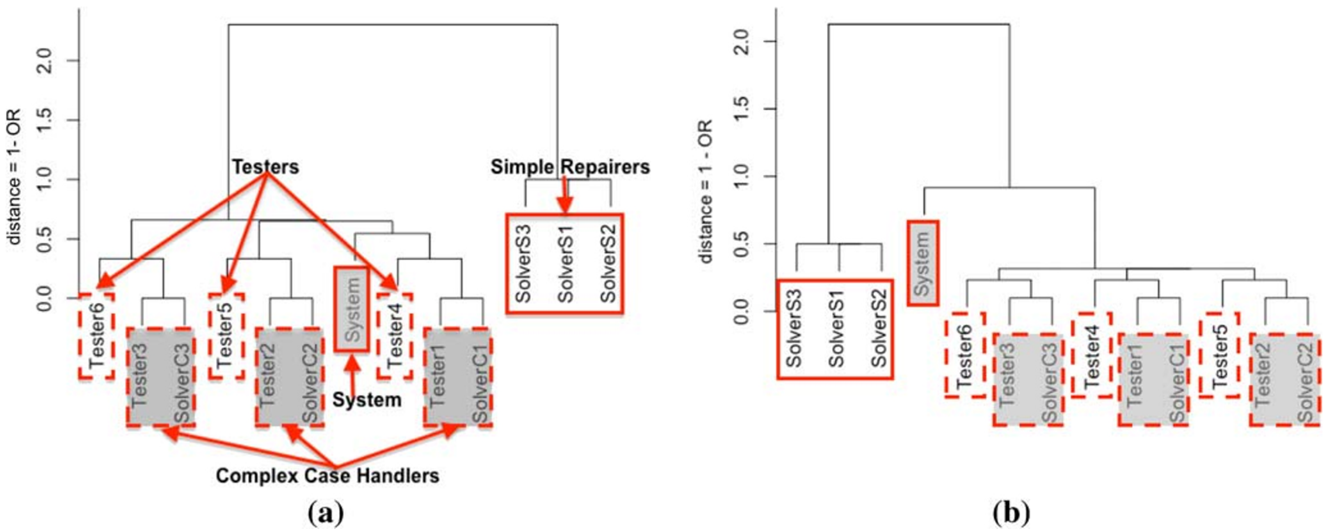


**Fig. 6** **a** Repair process: Mined organizational model from the event log**. b** Repair process: Manually constructed organizational model

matrix. Based on this matrix, the *n*-item frequent item (activity) sets are generated and used as candidate resource allocation rules. The next step for mining resource allocation rules is to logically link activities with originators. The organizational mining module within the *OrgMiner* links activities with originators through *AR* and *OR* computations. Further, *OR* also helps captures fine-grained characteristics of originators through their prior task completion data. A key challenge in resource allocation rule mining is that too many uninteresting rules are generated from the frequent itemsets, leading to inefficiency and low efficacy. In other words, post hoc handling of candidate rules is critical in resource allocation rule mining. We discuss this issue in the context of the repair process case below.

In case of the repair process event log, using the BPDA algorithm in the *OrgMiner* framework, we obtain the behavioral relation matrix for generating resource allocation rules. Using *minimum support* = 0.01 and *minimum confidence* = 0.1 thresholds for the behavioral relation matrix, 18,222 candidate rules are generated. Appendix 1 shows a sample of the candidate resource allocation rules generated. From the candidate rules generated, redundancy between several rules is evident. Traditionally, confidence (Agrawal and Srikant 1994) and lift (Kannan and Bhaskaran 2009) metrics from the fields of association rule learning as well as cooperation-based approach (Huang et al. 2012) from process mining field have been used to address the rule redundancy issue. Given the focus on mapping activities to appropriate resources, we use the *OR* metric used in the *OrgMiner* framework for filtering redundant rules instead of the traditional approaches. In our approach, the selected *k* value in the *top-k* method serves as a redundancy control knob for rule selection. The top-10 resource allocation rules generated are shown in Appendix 2. Lift, computed as $lift(x \rightarrow y) = \frac{conf(x \rightarrow y)}{supp(y)}$ is a correlation metric to filter uninteresting rules: a rule *r* is negatively correlated when *lift(r)* < 1 (Liu et al. 2012). Lift measures the degree of dependence between the itemsets. The higher lift values indicate the extent to which the rule is more likely to occur than if the antecedent and consequent in the rules were statistically independent. Such an understanding would inform decision-making for resource allocation.

The rules in Appendix 2 show actionable insights for resource allocation. As an example, rule # 1971 with lift 2.917 suggests that if the 'repair (simple)' activity, i.e., *Activity C1*, has been already attempted by two solvers in the Simple Repairers group (*SolverS1, SolverS2, SolverS3*), the task should be assigned to the remaining member of the group. This seems to logically make sense in that if prior attempts have been unsuccessful with some resources, another resource should be allocated to address the issue at hand. Similarly, the remaining rules are also actionable and are in accordance with the mined organizational model. Few points are noteworthy. First, by using the *OR* metric to filter the candidate rules, we can retain interesting (high lift), yet rare (low support) rules (e.g., rule # 1216 in Appendix 2), while ensuring high association between originators. Second, although association rules are commonly read as the presence of antecedent implying the presence of consequent, in this particular application the focus is on the mapping between activities and originators from underlying rules in the process model that are not explicit. As such, given the implicit policies in the latent process model, we do not consider the rules as direction sensitive, but rather emphasize association. The temporal nature of activities is accounted through the behavioral relations in the prior analysis step, which supports this rationale.

In addition to demonstrating the efficacy of the proposed approach based on the *OrgMiner* framework, we also assess the efficiency of the proposed approach in terms of least number of filtered rules generated that are relevant to the process context. In terms of the absolute computational time, the end-to-end analysis of the repair event log case study logged 9 min and 48 s on a machine with an Intel i7 CPU and 16 GB RAM. We also examine the number of filtered resource allocation rules generated under different minimum support and minimum confidence thresholds. We compare the results from the *OrgMiner* approach to the *Apriori* algorithm, interestingness-based filtering (lift) (Liu et al. 2012), and cooperation-based filtering (Huang et al. 2012) approaches.). We use a similar setup in comparing the approaches ($0.01 \leq$ support $\leq 0.1$, step = 0.01; $0.05 \leq$ confidence $\leq 0.5$, step = 0.05). The numbers of filtered rules generated are reported in Fig. 7(a) and (b). It is evident that our approach generates the tightest set of actionable rules as compared to other approaches.

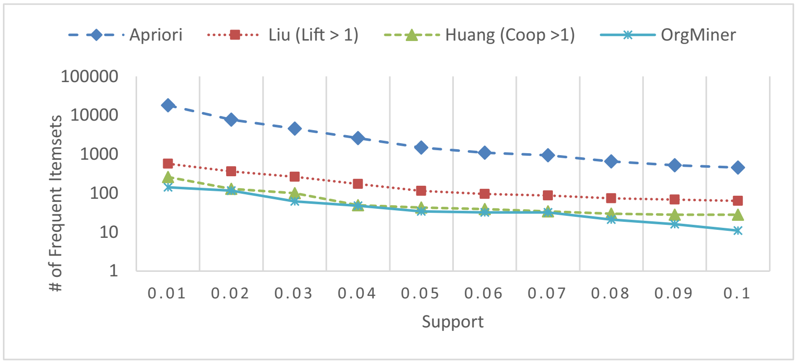
Further, we also examine the number of two-item rules, given that two-item rules are most relevant in a resource allocation context (Liu et al. 2012). The number of filtered two-item rules generated are reported in Fig. 8(a) and (b). In two-item rules case as well, it is shown that our approach generates the tightest set of filtered rules compared to other approaches. In sum, the feasibility and efficacy of the proposed approach, along with better rule generation efficiency, is evident for the resource allocation application.

## 6 Discussion

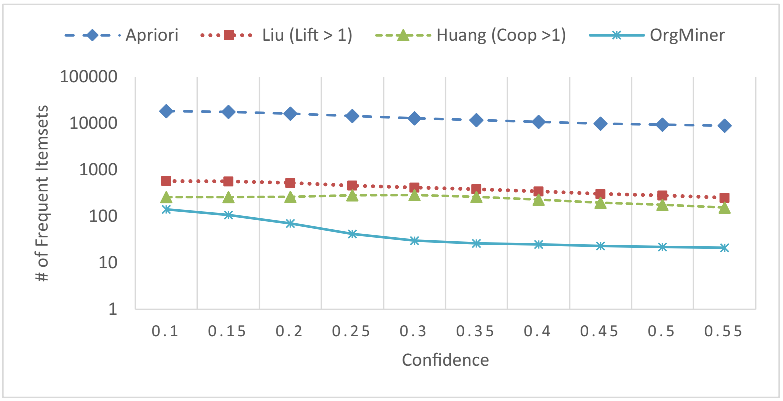
### 6.1 Related Work

As noted earlier, there are only a limited number of extant studies that have focused on the organizational PMA perspective (Ferreira and Alves 2012; Ni et al. 2011; Qiu and Lin 2011; Song and van der Aalst 2008). Review of the literature suggests that Song and van der Aalst's (2008) proposed comprehensive organizational mining approach is the first notable

**Fig. 7** **a** Number of filtered rules under different support thresholds (confidence = 0.1, minimum support = 0.01, step = 0.01). **b** Number of filtered rules under different confidence thresholds (support = 0.01, minimum confidence = 0.1, step = 0.05)



**(a) Number of filtered rules under different support thresholds (confidence = 0.1, minimum support = 0.01, step = 0.01)**



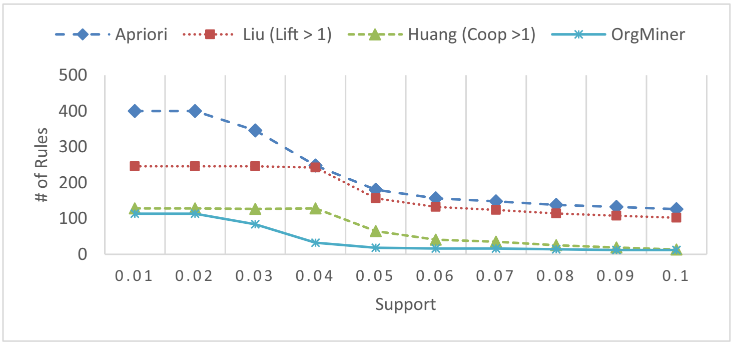
**(b) Number of filtered rules under different confidence thresholds (support = 0.01, minimum confidence = 0.1, step = 0.05)**

study in this area. Building on this study, Ferreira and Alves (2012) developed a user-community discovery approach with user interactions as the foundation for the clustering metrics. The approach proposed in our study extends both these approaches further. By leveraging behavioral patterns, we are not only able to consider user interactions, but also task similarities. In essence, with the help of event log aggregation techniques (Tao and Deokar 2015) as a method of conceptual modeling, the capabilities of task similarity-based analysis can be expanded to include relationships between originators indirectly involved in the process, thus supporting advanced analyses related to organizational mining (e.g. inter-organizational modeling, cross-functional mining, and domain modeling).

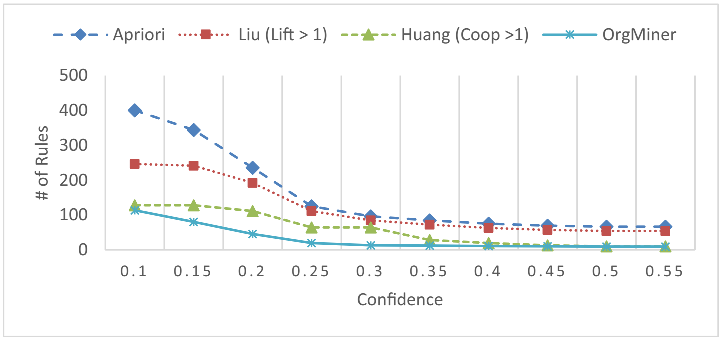
Ni et al. (2011) proposed a grid clustering based organizational mining approach that relies on the originator-task pairs. Compared to this approach, the proposed approach in this study is different in the following ways: 1) Instead of using single originator-task pairs, using behavior patterns is likely to better capture the interactions among users, particularly among users that are within the same functional area but across different organizational units (e.g. teams, departments). 2) In contrast to grid clustering, an AHC-based method can construct an organizational model at different levels that is more reflective of the actual organizational structure in practice, which in turn provides more information for model abstraction.

Qiu and Lin (2011) proposed an approach that enables learning an organizational model from social networks within organizations. This is slightly different compared to the aforementioned approaches since it relies on social network structure rather than event logs. Moreover, as stated by Qiu and Lin (2011), the learning algorithm is less effective because of the computational complexity of tree edit distances. Notably, we believe that employing novel, customized behavioral patterns based on the three basic ones proposed in our study can address this limitation. It can better portray the social network structure/communications and can significantly reduce the computational complexity through the use of the originator relatedness notion.

**Fig. 8** **a** Number of two-item rules under different support thresholds (confidence = 0.1, minimum support = 0.01, step = 0.01). **b** Number of two-item rules under different confidence thresholds (support = 0.01, minimum confidence = 0.1, step = 0.05)



**(a) Number of two-item rules under different support thresholds (confidence = 0.1, minimum support = 0.01, step = 0.01)**



**(b) Number of two-item rules under different confidence thresholds (support = 0.01, minimum confidence = 0.1, step = 0.05)**

## 6.2 Practitioner Implications

Organizational mining presents substantial value-building opportunities for practitioners in business process management. In particular, mined organizational models can be used to derive *job assignment rules, resource allocation rules*, or *user profiling rules*, across all three perspectives of PMA, namely *discovery, conformance checking*, and *enhancement*. Considering discovery as an example, the aforementioned rules are crucial for scenarios such as task assignment, resource allocation, and role-based access control. For instance, in a manuscript review process, editors should be able to select and assign reviewers to papers (task assignment), and reviewers should not have access to identifying user information if it is a double-blind review process (role-based access control). A number of studies have focused on researching issues in similar scenarios. Liu et al. (2008) proposed semi-automatic approaches for process-oriented staff assignment. Also, Liu et al. (2012) proposed an approach for generalizing resource allocation rules from event-based data. In a similar vein, Lohmann (2013) proposed an approach for role-based access control from the compliance assurance perspective, through analyzing event logs. Even though these approaches are based on mined organizational models as well, they rely heavily on the knowledge of pre-existing process models and/or organizational models. Relaxing this prerequisite in our research approach enables analysis for such scenarios, particularly where the underlying business process or organizational structure is unclear or too complex. A similar conclusion can be drawn from both conformance-checking and enhancement perspectives.

Further, the organizational mining approach proposed in this paper can be leveraged to analyze sensor networks, which are considered a backbone of Internet of Things (IoT). Clustering sensors, analogous to originators, of different functions into groups can potentially help improve the efficiencies of a sensor network, which is one of the foremost challenges in the IoT domain. For instance, service composition is used for orchestrating sensors/services in service-oriented architecture (SOA) based IoT (Garriga et al. 2018; Guinard et al. 2010; Vladimir et al. 2015). Existing query-based, recommendation-oriented service composition methods require detailed specification of services/sensors, whereas by clustering them functionally, service composition can be improved by augmenting existing methods.

## 7 Conclusion

In this study, we have proposed *OrgMiner*, a behavioral pattern-based framework for supporting decision-making from business process data. We have defined the concept of behavioral patterns, which rely on weak order relations appearing in event logs. Further, we identify and select behavioral patterns using appropriate metrics, and then apply them for organizational mining purposes. In this manner, organizational models containing clustered originators of activities in event logs can be built without explicit prior knowledge of process and/or organizational models. Through an empirical evaluation, the organizational models generated by *OrgMiner* are shown to be in accordance with expert-created manual models. Lastly, we demonstrated the utility of the framework for generating resource allocation rules, a typical organizational mining application.

We acknowledge some limitations of this study. To improve the accuracy of mined organizational models, further research is needed to develop a method to deal with the short loops in event logs. Also, we only considered the sequential relations between activities in event logs. Future work should consider incorporating more domain-specific knowledge (e.g. in the form of domain ontologies) that may be utilized to derive more complex relations. Domain-specific knowledge may be extracted from other organizational knowledge artifacts such as business rules or process models, if available. An example of such a technique is presented in a prior study (Tao and Deokar 2015). Furthermore, the identified behavioral patterns can be applied for other related PMA goals such as social network analysis, decision point mining, mining data flow and/or data object interactions, performance checking, and cost-benefit analysis. Thus, in addition to the control flow and organizational perspectives, we need to analyze the data entities traversing through business processes, which would provide a more comprehensive understanding of how originators of tasks are connected together (Sun et al. 2006). We also acknowledge that our work highly relies on the availability, meaningfulness, and correctness of the event logs. This implies requiring a pre-processing step including normalizing the event logs for ensuring their meaningfulness, merging event logs from different information systems together, and then preparing them for applying organizational mining approaches.

In sum, the proposed *OrgMiner* framework presents a viable and effective mechanism for analyzing event logs to infer organizational models. This behavioral patterns-based framework has been demonstrated in a real-world case study. We believe that the framework and its applications extend the understanding of the organizational mining aspect of process mining and analysis.

## Appendix 1. Repair Case: Top 10 Candidate Rules from the Event Log

| No | Rule | Support | Confidence | Lift |
|---|---|---|---|---|
| 3114 | {(B, Tester5), (C2, SolverS3), (E2, System)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |
| 5696 | {(C2, SolverS3), (D, Tester1), (D, Tester4)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |
| 6004 | {(C2, SolverS2), (D, Tester5), (D, Tester6)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |
| 6104 | {(C2, SolverS3), (D, Tester5), (D, Tester6)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |
| 8226 | {(B, Tester5), (C2, SolverS3), (E2, System), (F, System)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |
| 8231 | {(A, System), (B, Tester5), (C2, SolverS3), (E2, System)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |
| 8236 | {(B, Tester5), (C2, SolverS3), (E1, System), (E2, System)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |
| 11,210 | {(C2, SolverS3), (D, Tester1), (D, Tester4), (E2, System)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |
| 11,215 | {(C2, SolverS3), (D, Tester1), (D, Tester4),(F, System)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |
| 11,220 | {(A, System), (C2, SolverS3), (D, Tester1), (D, Tester4)} => {(C2, SolverS1)} | 0.01 | 1.00 | 5.348 |

## Appendix 2. Repair Case: Top 10 Resource Allocation Rules Filtered by OR

| No. | Rule | Support | Confidence | Lift |
|---|---|---|---|---|
| 1971 | {(C1, SolverS2),(C1, SolverS3)} = > {(C1, SolverS1)} | 0.012 | 0.545 | 2.917 |
| 1216 | {(D, Tester1),(D, Tester4)} = > {(B, Tester1)} | 0.01 | 0.417 | 2.408 |
| 269 | {(C1, SolverS1)} = > {(C1, SolverS2)} | 0.087 | 0.465 | 2.398 |
| 415 | {(D, Tester4),(D, Tester5)} = > {(B, Tester4)} | 0.011 | 0.344 | 2.338 |
| 1585 | {(D, Tester1),(D, Tester3)} = > {(B, Tester6)} | 0.01 | 0.333 | 1.852 |
| 304 | {(C2, SolverC1)} = > {(C2, SolverC3)} | 0.072 | 0.385 | 1.791 |
| 414 | {(B, Tester4),(D, Tester5)} = > {(D, Tester4)} | 0.011 | 0.333 | 1.642 |
| 3 | {(B, Tester4)} = > {(D, Tester4)} | 0.041 | 0.279 | 1.374 |
| 198 | {(B, Tester6)} = > {(D, Tester1)} | 0.051 | 0.283 | 1.288 |
| 140 | {(B, Tester1)} = > {(D, Tester5)} | 0.047 | 0.272 | 1.229 |

## References

Agrawal, R., and Srikant, R. (1994). "Fast Algorithms for Mining Association Rules," in *Proceeding VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487–499.

Alirezaei, E., and Parsa, S. (2018). "Adaptable cross-organizational unstructured business processes via dynamic rule-based semantic network," *Information Systems Frontiers*.

Alves de Medeiros, A. K., van Dongen, B., van der Aalst, W. M. P., and Weijters, A. J. M. M. (2004). "Process mining: Extending the alpha-algorithm to mine short loops," *Eindhoven*.

Becker, J., Delfmann, P., Dietrich, H. A., Steinhorst, M., & Eggert, M. (2016). Business process compliance checking – Applying and evaluating a generic pattern matching approach for conceptual models in the financial sector. *Information Systems Frontiers, 18*(2), 359–405. https://doi.org/10.1007/s10796-014-9529-y.

Bertolini, M., Bevilacqua, M., Ciarapica, F. E., & Giacchetta, G. (2011). Business process re-engineering in healthcare management: A case study. *Business Process Management Journal, 17*(1), 42–66. https://doi.org/10.1108/14637151111105571.

Bose, R., and van der Aalst, W. M. P. (2010). "Trace clustering based on conserved patterns: Towards achieving better process models," in *Business Process Management Workshops, Lecture Notes in Business Information Processing* (Vol. 43), pp. 170–181 (available at http://www.springerlink.com/index/n4h7t16v75752749.pdf).

Bose, R., & van der Aalst, W. M. P. (2012). Process Diagnostics Using Trace Alignment : Opportunities , Issues , and Challenges. *Information Systems, 37*(2), 117–141.

Bucher, T., Gericke, A., & Sigg, S. (2009). Process-centric business intelligence. *Business Process Management Journal, 15*(3), 408–429. https://doi.org/10.1108/14637150910960648.

Caron, F., Vanthienen, J., and Baesens, B. (2013). "Comprehensive rule-based compliance checking and risk management with process mining," *Decision Support Systems* (54:3), Elsevier B.V., pp. 1357–1369 (doi: https://doi.org/10.1016/j.dss.2012.12.012).

Cong, Z., Fernandez, A., Billhardt, H., & Lujak, M. (2015). Service discovery acceleration with hierarchical clustering. *Information Systems Frontiers, 17*(4), 799–808. https://doi.org/10.1007/s10796-014-9525-2.

Fan, S., Kang, L., & Zhao, J. L. (2015). Workflow-aware attention tracking to enhance collaboration management. *Information Systems Frontiers, 17*(6), 1253–1264. https://doi.org/10.1007/s10796-015-9565-2.

Ferreira, D. R., and Alves, C. (2012). "Discovering user communities in large event logs," *Lecture Notes in Business Information Processing* (99 LNBIP:PART 1), pp. 123–134 (doi: https://doi.org/10.1007/978-3-642-28108-2_11).

Ferreira, D. R., & Thom, L. H. (2012). A semantic approach to the discovery of workflow activity patterns in event logs. *International Journal of Business Process Integration and Management, 6*(1), 4–17.

Fowlkes, E. B., & Mallows, C. L. (1983). A method for comparing two hierarchical Clusterings. *Journal of the American Statistical Association, 78*(383), 553–569.

Fraga, A., Llorens, J., and Génova, G. (2019). "Towards a Methodology for Knowledge Reuse Based on Semantic Repositories," *Information Systems Frontiers* (21:1), Information Systems Frontiers, pp. 5–25 (doi: https://doi.org/10.1007/s10796-018-9862-7).

Garriga, M., De Renzis, A., Lizarralde, I., Flores, A., Mateos, C., Cechich, A., & Zunino, A. (2018). A structural-semantic web service selection approach to improve retrievability of web services. *Information Systems Frontiers, 20*(6), 1319–1344. https://doi.org/10.1007/s10796-016-9731-1.

Ghattas, J., Soffer, P., and Peleg, M. (2014). "Improving business process decision making based on past experience," *Decision Support Systems* (59), Elsevier B.V., pp. 93–107 (doi: https://doi.org/10.1016/j.dss.2013.10.009).

Guillet, F., and Hamilton, H. J. (Eds.). (2007). *Quality Measures in Data Mining, Vol. 43*, Springer.

Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., and Savio, D. (2010). "Interacting with the SOA-Based Internet of Things : Discovery , Query , Selection , and On-Demand Provisioning of Web Services," *Services Computing,* IEEE Transactions on (3:3), pp. 223–235.

Günther, C. W., and van der Aalst, W. M. P. (2007). "Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics," in *Business Process Management, Lecture Notes in Computer Science* (Vol. 4714), pp. 328–343.

Huang, Z., Lu, X., and Duan, H. (2012). "Resource behavior measure and application in business process management," *Expert Systems with Applications* (39:7), Elsevier Ltd, pp. 6458–6468 (doi: https://doi.org/10.1016/j.eswa.2011.12.061).

IEEE Task Force on Process Mining. (2011). "Process Mining Manifesto, " in *Business Process Management Workshops, Lecture Notes in Business Information Processing* (Vol. 99), pp. 169–194.

Jareevongpiboon, W., & Janecek, P. (2013). Ontological approach to enhance results of business process mining and analysis. *Business Process Management Journal, 19*(3), 459–476. https://doi.org/10.1108/14637151311319905.

Kannan, S., & Bhaskaran, R. (2009). Association rule pruning based on interestingness measures with clustering. *Journal of Computer Science, 6*(1), 35–43 available at http://arxiv.org/abs/0912.1822.

Kluza, K., and Nalepa, G. J. (2018). "Formal model of business processes integrated with business rules," Information Systems Frontiers, Information Systems Frontiers, pp. 1–19 (doi: https://doi.org/10.1007/s10796-018-9826-y).

Köck, M., & Paramythis, A. (2011). Activity sequence modelling and dynamic clustering for personalized e-learning. *User Modeling and User-Adapted Interaction, 21*(1–2), 51–97. https://doi.org/10.1007/s11257-010-9087-z.

Leyer, M., Schneider, C., and Claus, N. (2016). "Would you like to know who knows? Connecting employees based on process-oriented knowledge mapping," *Decision Support Systems* (87), Elsevier B.V., pp. 94–104 (doi: https://doi.org/10.1016/j.dss.2016.05.003).

Liu, Y., Wang, J., Yang, Y., & Sun, J. (2008). A semi-automatic approach for workflow staff assignment. *Computers in Industry, 59*(5), 463–476. https://doi.org/10.1016/j.compind.2007.12.002.

Liu, T., Cheng, Y., and Ni, Z. (2012). "Mining event logs to support workflow resource allocation," *Knowledge-Based Systems* (35), Elsevier B.V., pp. 320–331 (doi: https://doi.org/10.1016/j.knosys.2012.05.010).

Lohmann, N. (2013). Compliance by design for artifact-centric business processes. *Information Systems, 38*(4), 606–618. https://doi.org/10.1016/j.is.2012.07.003.

Nguyen, D., Vo, B., and Le, B. (2014). "Efficient strategies for parallel mining class association rules," *Expert Systems with Applications* (41:10), Elsevier Ltd, pp. 4716–4729 (doi: https://doi.org/10.1016/j.eswa.2014.01.038).

Ni, Z., Wang, S., and Li, H. (2011). "Mining organizational structure from workflow logs," in *e-Education, Entertainment and e-Management (ICEEE), 2011 International Conference on*, pp. 222–225 (available at http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6137791).

Pourmasoumi, A., Kahani, M., and Bagheri, E. (2017). "Mining variable fragments from process event logs," *Information Systems Frontiers* (19:6), Information Systems Frontiers, pp. 1423–1443 (doi: https://doi.org/10.1007/s10796-016-9662-x).

Qiu, J., and Lin, Z. (2011). "A framework for exploring organizational structure in dynamic social networks," *Decision Support Systems* (51:4), Elsevier B.V., pp. 760–771 (doi: https://doi.org/10.1016/j.dss.2011.01.011).

Savasere, A., Omiecinski, E., and Navathe, S. (1995). "An efficient algorithm for mining association rules in large databases," (available at http://smartech.gatech.edu/handle/1853/6678).

Sellami, R., Gaaloul, W., & Defude, B. (2013). Process socio space discovery based on semantic logs. *Journal of Internet Technology, 14*(3), 401–412. https://doi.org/10.6138/JIT.2013.14.3.05.

Shepitsen, A., Gemmell, J., Mobasher, B., and Burke, R. (2008). "Personalized recommendation in social tagging systems using hierarchical clustering," in *Proceedings of the 2008 ACM conference on Recommender systems - RecSys '08*, New York, New York, USA: ACM Press, p. 259 (doi: https://doi.org/10.1145/1454008.1454048).

Smirnov, S., Weidlich, M., Mendling, J., & Weske, M. (2012). Action patterns in business process model repositories. *Computers in Industry, 63*(2), 98–111.

Song, M., and van der Aalst, W. M. P. (2008). "Towards comprehensive support for organizational mining," *Decision Support Systems* (46:1), Elsevier B.V., pp. 300–317 (doi: https://doi.org/10.1016/j.dss.2008.07.002).

Sun, S. X., & Zhao, J. L. (2013). Formal workflow design analytics using data flow modeling. *Decision Support Systems, 55*(1), 270–283. https://doi.org/10.1016/j.dss.2013.01.028.

Sun, S. X., Zhao, J. L., Nunamaker, J. F., & Sheng, O. R. L. (2006). Formulating the data-flow perspective for business process management. *Information Systems Research, 17*(4), 374–391. https://doi.org/10.1287/isre.1060.0105.

Tan, W., Jiang, C., Li, L., & Lv, Z. (2008). Role-oriented process-driven enterprise cooperative work using the combined rule scheduling strategies. *Information Systems Frontiers, 10*(5), 519–529. https://doi.org/10.1007/s10796-008-9107-2.

Tao, J., & Deokar, A. V. (2015). "Semantics-based Event Log Aggregation for Process Mining and Analytics," *Information Systems Frontiers,* (17):1209–1226. https://doi.org/10.1007/s10796-015-9563-4.

Thomas, O., and Fellmann, M. (2006). "Semantic event-driven process chains," in *Proceedings of the Workshop on Semantics for Business Process Management (SBPM '06), held at the 3rd European Semantic Web Conference (ESWC 2006)*, Budva, Montenegro, June, p. 2.

Tiwari, A., Turner, C. J., & Majeed, B. (2008). A review of business process mining: State-of-the-art and future trends. *Business Process Management Journal, 14*(1), 5–22. https://doi.org/10.1108/14637150810849373.

van der Aalst, W. M. P. (2011). *Process mining: Discovery, conformance and enhancement of business processes (2nd ed.)*. Berlin Heidelberg: Springer. https://doi.org/10.1007/978-3-662-49851-4.

van der Aalst, W. M. P. (2012a). Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems, 3*(2), 1–17. https://doi.org/10.1145/2229156.2229157.

van der Aalst, W. M. P. (2012b). *"Process Mining," Communications of the ACM (55:8)* (pp. 76–83). Berlin: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19345-3.

van der Aalst, W. M. P., Alves de Medeiros, A. K., & Weijters, A. J. M. M. (2005). In G. Ciardo & P. Darondeau (Eds.), *"Genetic process mining," in Applications and Theory of Petri Nets 2005, Proceedings* (Vol. 3536, pp. 48–69). Berlin: Springer-Verlag.

van der Aalst, W. M. P., Zhao, J. L., & Wang, H. J. (2015). Business process intelligence: Connecting data and processes. *ACM Transactions on Management Information Systems, 5*(4), 1–7. https://doi.org/10.1145/2685352.

van Dongen, B., & van der Aalst, W. M. P. (2004). EMiT: A process mining tool. In *Applications and Theory of Petri Nets 2004, Proceedings* (Vol. 3099, pp. 454–463). Berlin: Springer-Verlag Berlin.

Vladimir, K., Budiselić, I., & Srbljić, S. (2015). Consumerized and peer-tutored service composition. *Expert Systems with Applications, 42*(3), 1028–1038. https://doi.org/10.1016/j.eswa.2014.09.033.

Wahyudi, A., Kuk, G., and Janssen, M. (2018). "A Process Pattern Model for Tackling and Improving Big Data Quality," *Information Systems Frontiers* (20:3), Information Systems Frontiers, pp. 457–469 (doi: https://doi.org/10.1007/s10796-017-9822-7).

**Amit V. Deokar** is an Associate Professor of Management Information Systems in the Manning School of Business at the University of Massachusetts Lowell. He received his PhD in Management Information Systems from the University of Arizona. He also earned a MS in Industrial Engineering from the University of Arizona and a BE in Mechanical Engineering from VJTI, University of Mumbai. His research interests include data analytics, enterprise data management, business intelligence, business process management, and collaboration processes. His work has been published in journals such as Journal of Management Information Systems, Decision Support Systems (DSS) and Information Systems Frontiers. He is the editor of the e-Service Journal, and also a member of the editorial board of DSS and BPMJ journals. He has served in various executive roles for the AIS Special Interest Group on Decision Support and Analytics (SIGDSA) and the Midwest AIS Chapter. He was recognized with the 2014 IBM Faculty Award for his research and teaching in the areas of analytics and big data.

**Jie Tao** is an Assistant Professor of Information Systems and Operations Management in the Charles F. Dolan School of Business at Fairfield University. Dr. Tao received his doctoral degree in Information Systems from Dakota State University. His recent research interests mainly include data analytics and business process management. He received the AIS ATLAS award for his service to AIS in the year 2013. Dr. Tao is also a Nvidia Deep Learning Institute certified instructor.