Information Systems 88 (2020) 101463

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Retrieving the resource availability calendars of a process from an event log

Niels Martin [a,b,*], Benoît Depaire [a], An Caris [c], Dimitri Schepers [a]

[a] Hasselt University, Research group Business Informatics, Martelarenlaan 42, 3500 Hasselt, Belgium
[b] Research Foundation Flanders (FWO), Egmontstraat 5, 1000 Brussels, Belgium
[c] Hasselt University, Research group Logistics, Martelarenlaan 42, 3500 Hasselt, Belgium

ARTICLE INFO

Article history:
Received 11 January 2019
Received in revised form 12 October 2019
Accepted 25 October 2019
Available online 31 October 2019
Recommended by Stephanie Rinderle-Ma

Keywords:
Resource availability calendars
Human resources availability
Process mining
Event logs
Process execution data

ABSTRACT

Knowing the availability of human resources for a business process is required, e.g., when allocating resources to work items, or when analyzing the process using a simulation model. In this respect, it should be taken into account that staff members are not permanently available and that they can be involved in multiple processes within the company. Consequently, it is far from trivial to specify their availability for the single process from, e.g., generic timetables. To this end, this paper presents a new method to automatically retrieve resource availability calendars from event logs containing process execution information. The retrieved resource availability calendars are the first to take into account (i) the temporal dimension of availability, i.e. the time of day at which a resource is available, and (ii) intermediate availability interruptions (e.g. due to a break). Empirical evaluation using synthetic data shows that the method's key outputs closely resemble their equivalents in reality.

1. Introduction

Human resources play a pivotal role in many business processes as they ensure that cases (e.g. patients or files) can move ahead in the process. When, e.g., allocating resources to work items, or when analyzing the process using a simulation model, it is important to know the availability of human resources. In this respect, it needs to be recognized that staff members are not permanently available [1,2] and tend to be involved in multiple processes [3,4]. When, e.g., a process analysis project focuses on a single process (e.g. the inbound logistics process or the billing process), the availability of resources for that particular process needs to be determined. Resource availability can be expressed using resource availability calendars, marking time periods during which a resource can be allocated to work items in the process under analysis.

While there exists a rich body of literature to optimally create resource timetables [5–8] to, e.g., minimize staffing costs while maintaining service levels, there is strikingly little work on empirically deducing the resource availability prevailing in reality. Timetables from, e.g., the human resources department are a straightforward information source, but typically express the 'formal' availability of a staff member for the company as a whole. To capture actual resource availability for a single process, which is the focus of this paper, one can resort to interviews and observations. However, staff might find it difficult to articulate the time dedicated to a particular process [9] and extensive observations are highly time-consuming and might influence behavior [10]. Consequently, resource availability is often modeled based on incorrect assumptions [4].

This paper addresses the problem of discovering resource availability calendars, which express the availability of resources for a single process from empirical data. A method is proposed which is inspired by process mining. Process mining encompasses techniques to automatically extract process models from process execution data files called event logs [11]. Despite the wide presence of systems generating event logs (e.g. Enterprise Resource Planning Systems and Hospital Information Systems), no method is available to automatically retrieve resource availability calendars from these logs. Currently, resource availability research in process mining is limited to quantifying aggregate metrics such as the percentage of time that a resource is available.

The data-driven approach defined in this paper is the first to retrieve resource availability calendars which take into account (i) the temporal dimension of availability, i.e. the time of day at which a resource is available, and (ii) intermediate availability interruptions, i.e. time periods during which the availability for the process under consideration is interrupted due to, e.g., a break or the allocation of a resource to another process. The retrieved availability calendar can, for example, express that on Monday, a resource is available for the process under consideration from 08:30 until 10:30, from 10:50 until 12:00, and from 14:00 until

* Corresponding author at: Hasselt University, Research group Business Informatics, Martelarenlaan 42, 3500 Hasselt, Belgium.
E-mail address: niels.martin@uhasselt.be (N. Martin).

https://doi.org/10.1016/j.is.2019.101463
0306-4379/© 2019 Elsevier Ltd. All rights reserved.

15:30. The availability calendars provide rich insights in resource availability patterns and can be directly integrated into, e.g., a process simulation model. In this way, the paper complements existing work on resource scheduling, assignment, and allocation with an inductive data-driven approach. Moreover, it advances research on resource availability modeling in the process mining field. As will be detailed later, the current work constitutes a significant extension of prior work on this topic by the authors [12]. The proposed approach is fully implemented in the programming language R[1] and is evaluated using synthetic data. The evaluation focuses on both the accuracy of the retrieved availability at day level and the effect of incorporating data-driven availability calendars in a process simulation model. The empirical results demonstrate that the method's key outputs closely resemble their equivalents in reality. This shows that the proposed method constitutes a valuable tool to gain data-driven insights in the availability of a resource for a particular process.

The remainder of this paper is structured as follows. Section 2 discusses some terminology, provides an overview of the related work, and presents a running example. Section 3 outlines the data-driven method to retrieve resource availability calendars. This method is empirically evaluated in Section 4. The paper ends with a discussion in Section 5, and a conclusion in Section 6.

2. Preliminaries

2.1. Concepts

To position the notion of a resource availability calendar, a distinction is made between the terms resource scheduling, resource assignment, and resource allocation. *Resource scheduling* reflects the development of timetables for staff members, showing when they have to work, to ensure that the company's demand is satisfied [5,7]. While resource scheduling focuses on determining the required workforce during a time period, resource assignment and allocation need adequate specifications in, e.g., a workflow engine. *Resource assignment* specifies, at design time, conditions that a staff member needs to fulfill to potentially execute a particular activity. Given the resource assignment, *resource allocation* determines which staff member will actually perform a work item (i.e. the execution of an activity for a particular case) [13].

This paper presents a method to create resource availability calendars, which reflect a resource's availability for a particular business process, i.e. it expresses timeframes during which a resource can be allocated to activities in that process. Several meanings have been attributed to the concept of *availability* in literature. Huang et al. [14] and Zhao et al. [15] perceive a resource as being available when it is idle and can accept new work items. Conversely, in this paper, a resource is available when he/she can be allocated to work items in the process, regardless of whether it is idle or not. This is consistent with Nakatumba [1] and van der Aalst et al. [4], where resource availability expresses the percentage of time during which a resource can potentially perform work items. However, this paper will also consider the temporal dimension of availability and intermediate interruptions.

2.2. Related work

To position the proposed method with respect to the state of the art, this subsection provides an overview of the key related work. To structure the discussion, prior research is subdivided in five streams: (i) the development of resource schedules, (ii) the assignment of resources, (iii) the allocation of resources, (iv) the specification of resource availability in process simulation models, and (v) the retrieval of resource availability insights from an event log.

[1] https://www.r-project.org.

2.2.1. Resource schedule development

Within the operations research field, extensive research has been done on staff scheduling problems. These techniques aim to determine staff members' timetables, which will determine how many employees are available at a particular point in time [5,7]. Staff schedules are determined given an objective function aiming to, for example, minimize staffing costs or minimize service waiting times. Feasible schedules need to fulfill a set of constraints, which can relate to, e.g., guaranteeing that the workload is covered, and ensuring that the number of consecutive days that a staff member should work is limited [6]. Solution techniques include mathematical optimization methods such as integer linear programming or heuristics such as simulated annealing [7]. Inductive methods, such as the one presented in this paper, can be highly valuable in shaping the constraints in a staff scheduling problem starting from real-life data. For further reading on staff scheduling from an operations research perspective, the reader is referred to review papers such as Ernst et al. [5], Van den Bergh et al. [6], De Bruecker et al. [7], and Defraeye and Van Nieuwenhuyse [8].

2.2.2. Resource assignment

As highlighted in Section 2.1, resource assignment determines conditions that an employee should fulfill to potentially perform an activity [13]. Resource assignment can be linked to the workflow resource patterns defined by Russell et al. [16]. In their work, Russell et al. [16] distinguish 43 patterns to express how resources are presented and used in processes. These patterns are subdivided into seven categories: creation, push, pull, detour, auto-start, visibility, and multiple resource patterns [16]. Resource assignment can be related to the creation patterns [17].

Significant research has been done on the assignment of resources to activities. The resource assignment problem is studied taking into account different conditions such as the presence of a task deadline [18], cooperation between resources [19], and the structure of the process to which resources are assigned [20]. Other authors aim to detect rules underlying resource assignment using techniques such as decision tree analysis [21,22], association rule mining [23], naive Bayes classifiers [22], support vector machines [22], and declarative mining [24].

Even though resource availability will impose constraints on resource assignment, many of the aforementioned works attribute no attention to resource availability [20,21,23–25]. Other authors briefly mention resource availability. For instance: Kumar et al. [19] assume that resources are permanently available. Kumar et al. [18] integrate resource availability by subjectively considering the resource's planned and unplanned absence and its workload. An alternative approach is used in Liu et al. [22], where a supervisor should decide upon the feasibility of the resource assignment based on, amongst others, resource availability. The inductive approach proposed in this paper is complementary to existing work on resource assignment. Data-driven insights can be a valuable alternative to the aforementioned assumptions related to the availability of a resource for a particular process.

2.2.3. Resource allocation

Taking into consideration the resource assignment, resource allocation determines which specific employee will execute a work item [13]. Within the context of the workflow resource patterns [16], resource allocation can be linked to push and pull patterns [17].

Resource allocation has also been extensively studied [26]. This involves investigating the allocation of work items to specific staff members while taking into account, for example, relations between resources [27] and emergency circumstances [28]. Senkul and Toroslu [29] present an architecture to incorporate resource allocation constraints while distributing work items amongst resources. An example of a resource allocation constraint is that if a particular activity is executed by a specific resource for a case, another activity has to be executed by the same resource for that case [29].

Within the workflow management system research domain, Mans et al. [30] extend a system providing work lists to staff members with appointment creation functionalities. In their proposed conceptual model, they introduce a calendar component that fills up available time slots. Havur et al. [26] extend the work by Mans et al. [30] and also take resource dependencies and conflicts in consideration during resource allocation. In this respect, they explicitly state that a resource calendar, expressing amongst others its availability and unavailability, has to be accessible for resource allocation purposes [26]. Data-driven insights on resource availability can support the configuration of such a calendar. Barba et al. [31] propose a system to generate enactment plans for business processes, which plan work items while taking into account both the process control-flow (i.e. the order of activities) and the resource perspective. Resources becoming available or unavailable can lead to updates of such enactment plans [31]. Approaches such as the one proposed in this paper can be leveraged to take into account resource availability when conceiving enactment plans for business processes.

2.2.4. *Resource availability specification in process simulation models*

In the introduction, it is highlighted that the development of a process simulation model requires the specification of the resource availability. A process simulation model is a computer model that can be used to identify the effects of operational changes in the process before they are actually implemented [32–34]. The development of a simulation model requires the definition of resource availability for the process under consideration [35].

When simulation case studies are presented in literature, the way in which resource availability is modeled is often not explicitly mentioned (e.g. in [36] and [37]). When it is reported, resource availability specifications often originate from domain expert input (e.g. in [38]). The aforementioned studies originate from environments which are very structured in terms of resource presence (e.g., due to the use of fixed timetables) such as a hospital [38], an airport [36] or an assembly line [37]. Other process contexts are less structured. Leyer and Moormann [39], for instance, simulate a loan application process. All information regarding staff availability and their availability percentage to the process originates from a domain expert. In processes where staff members have significant freedom to divide their attention between processes, it is far from trivial for domain experts to make resource availability for one particular process explicit. In this respect, a data-driven approach such as the one presented in this paper is valuable.

2.2.5. *Resource availability retrieval using an event log*

Related to studying resource availability using an event log, the core topic of this paper, existing work is scarce. Wombacher et al. [40] focus on the activity duration, but also retrieve the start and end of a working day from an event log. These coincide with the start of the first activity executed by the resource, and the completion of the last one on a particular day, respectively [40]. Similarly, Nakatumba [1] equates a resource's available period to the elapsed time between the start of the first activity instance associated to that resource, and the end of the last one during a time horizon such as a day. As a consequence, a resource is assumed to be permanently available within this period as, e.g., breaks are not considered. Using the length of the availability period, an availability measure is calculated by dividing the cumulative active time by the length of the availability period [1]. However, calling this a resource availability measure can be misleading as it actually represents the active time instead of the time that the resource is available for the process.

Consistent with Nakatumba [1], Reijers et al. [41] also suggest to express resource availability as a percentage of time to accommodate for the involvement of resources in multiple processes. The same holds for Leyer and Moormann [39], where a resource's presence for a particular process is defined as a percentage of the total availability. However, all these approaches do not take into account the temporal dimension of availability, i.e. the time of day at which a resource is available. When specifying resource availability in, e.g., a simulation model, this temporal dimension is important as it can influence performance measures such as the cycle time of a case (i.e. the elapsed time between the start and the end of the process execution for a case [42]). This stresses the need to extend the state of the art in process mining to retrieve the resource availability calendars of a process from an event log.

2.3. *Running example*

Throughout this paper, a running example is used to illustrate the introduced concepts and methods. To this end, a fictitious business process consisting of seven activities is used, which is visualized as a Business Process Model and Notation (BPMN) model in Fig. 1. BPMN is a standard notation to model business processes, with the diamond-shapes containing label "x" representing exclusive choice structures [43]. Abstract activity labels are purposefully used to stress the generality of the proposed method. However, one might position the process within an administrative context such as the process of judging an environmental permit request. Activity duration is expressed in minutes and follows a triangular distribution. Case interarrival times, expressing the time that elapses between the arrival of subsequent cases in the process [35], are assumed to follow an exponential distribution with a mean of six minutes. Fig. 1 is annotated with all assumed parameters, including the resources that can execute a particular activity. For illustrative purposes, resource assignment is person-based, i.e. one or more specific resources are linked to an activity. In practice, role-based resource assignment often prevails, implying that a resource role (e.g. administrative clerk, team lead, manager) is linked to an activity. However, the prevailing resource assignment approach will not influence the applicability of the proposed method. Besides the annotated parameters in Fig. 1, it should be noted that each activity has a queue where cases wait to be served by a resource.

To analyze the process, company management wants to evaluate several policy alternatives using process simulation. When building a simulation model, the specification of resource availability calendars is one of the key modeling tasks. For this purpose, the company will follow a data-driven approach. To this end, the event log generated by the system supporting the process in Fig. 1 will be used. To illustrate the event log structure, Table 1 provides an event log excerpt. Each line represents an event, i.e. 'something' that happened in the process. For example: the first line shows that the execution of activity B by resource r2 on case 72 (e.g. file, order, patient, …) started on April 6th at 08:50:22. The same activity execution ended at 08:56:57, as shown in the second line of Table 1.

Resource availability calendar specification in a process simulation model will be the use case in this paper. However, this does not limit the generic contribution of the proposed method. From the related work section, it follows that inductive resource availability insights can, for instance, also be leveraged to specify constraints when solving staff scheduling problems. Moreover, the obtained output provides a rich basis for the calculation of resource availability metrics, which are useful for performance management purposes.

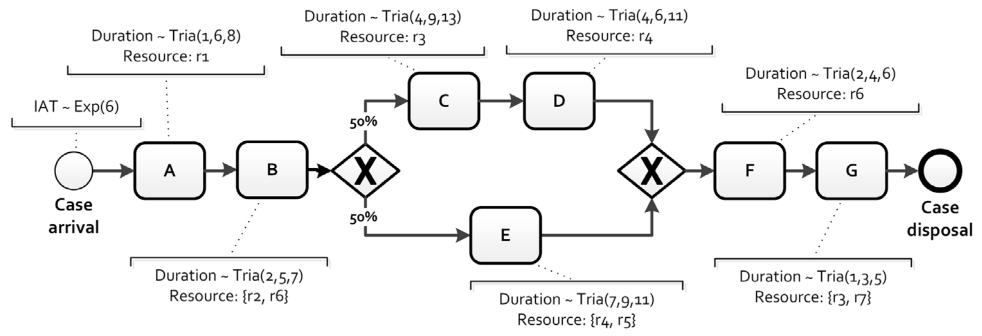


Fig. 1. Process model of the running example.

Table 1
Illustration of event log structure.

Case id	Timestamp	Activity	Event type	Resource
...	...	...	...	...
72	06/04/2018 08:50:22	B	Start	r2
72	06/04/2018 08:56:57	B	Complete	r2
72	06/04/2018 09:02:01	C	Start	r3
56	06/04/2018 09:05:51	D	Start	r4
72	06/04/2018 09:08:42	C	Complete	r3
38	06/04/2018 09:08:42	G	Start	r3
56	06/04/2018 09:11:17	D	Complete	r4
64	06/04/2018 09:11:17	D	Start	r4
38	06/04/2018 09:11:52	G	Complete	r3
76	06/04/2018 09:14:21	B	Start	r2
76	06/04/2018 09:20:07	B	Complete	r2
64	06/04/2018 09:20:41	D	Complete	r4
64	06/04/2018 09:20:58	F	Start	r6
64	06/04/2018 09:24:02	F	Complete	r6
...	...	...	...	...

3. Data-driven resource availability calendar retrieval method

This section outlines the data-driven resource availability calendar retrieval method, which takes an event log as an input, and generates resource availability calendars as an output. These availability calendars convey inductive insights in resource availability patterns and can be directly integrated into, e.g., a process simulation model. As shown in Fig. 2, the method consists of two stages. In the first stage, the event log is converted to an activity log in which each entry presents an activity instance (i.e. the execution of an activity on a case). This is done by mapping start events to their corresponding completion events. Moreover, the arrival time of a case at an activity (i.e. the time at which a case joins the queue of that activity) is added to create an arrival time imputed activity log. This log is used to retrieve daily availability records, expressing the availability of a particular resource on a particular day for the process under consideration. In the second stage, these daily availability records are leveraged to generate resource availability calendars using sampling methods. Two approaches are proposed: direct sampling and cluster-based sampling.

The method to retrieve daily availability records from an event log, i.e. the first stage in Fig. 2, is conceptually introduced in Martin et al. [12]. This paper significantly extends the latter conference contribution by formalizing the key concepts used in the first stage and by empirically evaluating the accuracy of its results. Moreover, the prior work is extended with the second stage in Fig. 2, enabling the definition of resource availability calendars having an empirical basis.

The remainder of this section details the method. Section 3.1 outlines the event log requirements. Section 3.2 discusses and formalizes the first stage, i.e. the creation of an arrival time imputed activity log (Section 3.2.1), the extraction of daily availability records (Section 3.2.2) and daily availability record post-processing (Section 3.2.3). Section 3.3 introduces the second stage, i.e. the creation of resource availability calendars.

3.1. Event log requirements

The example in Table 1 illustrates the minimal event log requirements to retrieve resource availability calendars using the method presented in this paper. Besides the case and activity to which an event is related, the timestamp, execution resource and transaction type also need to be recorded. Two transaction types have to be registered: start and complete. These event log requirements are formalized in Definition 1 using the notation introduced by van der Aalst [11].

Definition 1 (*Event Log Requirements*)**.** An event log E contains a set of events e. Each event e represents the start or completion of the execution of an activity a by resource r on case c at time τ. Hence, each event e is represented as a tuple e = (c, a, r, τ, φ) with:

- $\#_c(e)$ representing the case associated to event e
- $\#_a(e)$ representing the activity associated to event e
- $\#_r(e)$ representing the resource associated to event e
- $\#_\tau(e)$ representing the timestamp associated to event e
- $\#_\varphi(e)$ representing the transaction type associated to event e

Then, $\forall e \in E : \#_c(e) \neq \perp \wedge \#_a(e) \neq \perp \wedge \#_r(e) \neq \perp \wedge \#_\tau(e) \neq \perp \wedge \#_\varphi(e) \in \{start, complete\}$, where $\perp$ represents a null value. Moreover, every start event should have an accompanying complete event, i.e. $\forall e_1 \in E, \exists e_2 \in E : \#_c(e_1) = \#_c(e_2) \wedge \#_a(e_1) = \#_a(e_2) \wedge \#_r(e_1) = \#_r(e_2) \wedge \#_\tau(e_1) \leq \#_\tau(e_2) \wedge \#_\varphi(e_1) = start \wedge \#_\varphi(e_2) = complete$. When $|e_2| > 1$, i.e. when more than one event satisfies the conditions outlined for e_2, it is required that $|e_1| = |e_2|$.

3.2. Stage 1: Daily availability record retrieval

3.2.1. Arrival time imputed activity log creation

The method to retrieve daily availability records uses an arrival time imputed activity log as an input. This requires (i) the conversion of the event log to an activity log and (ii) the imputation of arrival times. The first step, i.e. the conversion to an activity log, introduces the notion of an activity instance. To this end, each start event is mapped to its associated completion event, which is the complete event related to the same case and activity. For instance: the first and third row in the event log in Table 1 are combined in a single row in the activity log. When multiple start and complete events are present for a combination of a case, activity and resource, the first occurring start event will

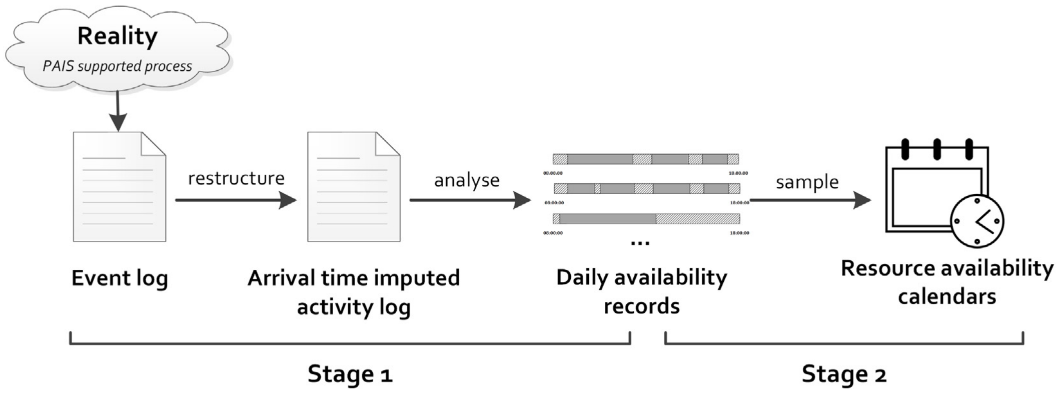

Reality
PAIS supported process
restructure
analyse
sample
Event log
Arrival time imputed activity log
Daily availability records
Resource availability calendars
Stage 1
Stage 2

Fig. 2. Overview of the proposed method.

iteratively be mapped to the first occurring unmapped complete event. For more complex mappings, techniques such as the ones described by Baier et al. [44] can be used.

The second conversion step requires the imputation of the arrival time of a case at an activity. This arrival time expresses the time at which a case joins the queue of an activity and, hence, requests service from its resource(s). When arrival times are recorded in an event log, as is the case in a Q-log [45], they can just be included in the first conversion step and no further efforts are required. When, to the contrary, arrival times are unknown, they can be imputed using a suitable heuristic. An example of a simple heuristic is to approximate a case's arrival time at an activity by the completion time of the preceding activity. In the running example, this would imply that arrival at activity B can be proxied by the completion of A. This requires knowledge on the prior activity, which is a control-flow notion. The latter notion can be gathered by applying an existing control-flow discovery algorithm to the event log. As demonstrated by the ongoing research on control-flow discovery, this is not trivial for complex processes. As control-flow discovery is beyond the scope of this paper, the reader is referred to, e.g.,van der Aalst [11], De Weerdt et al. [46], and Augusto et al. [47] for more background on this topic. Note that the imputation of arrival times is the only step in the proposed method where a control-flow notion is leveraged. The remainder of the method is independent of the control-flow complexity.

The notion of an arrival time imputed activity log is formalized in Definition 2. For illustrative purposes, the arrival time imputed activity log obtained after converting the event log in Table 1 is shown in Table 2. Each line represents an activity instance and contains the arrival time of the case at the activity, the start time of activity execution and the time at which activity execution is completed.

Definition 2 (*Arrival Time Imputed Activity Log*)**.** Let L_a be an arrival time imputed activity log based on event log E. Then L_a is composed of a set of activity instances η. Each activity instance η depicts the execution of an activity a by resource r on case c, where c arrives at a at $\tau_{arrival}$, the execution of a starts at time τ_{start} and is completed at time $\tau_{complete}$. Hence, an activity instance is represented as a tuple $\eta = (c, a, r, \tau_{arrival}, \tau_{start}, \tau_{complete})$, where $\#_k(\eta)$ represents the value of attribute k for activity instance η as suggested for events in Definition 1.

3.2.2. *Daily availability record retrieval*

The arrival time imputed activity log is the input for the retrieval of daily availability records. As highlighted earlier, a daily availability record expresses the availability of a resource for a process on a particular day. Hence, it divides a particular day in time intervals during which the resource is either available or unavailable for the process. This is formalized in Definition 3.

Definition 3 (*Daily Availability Record*)**.** Let $D_{r,d}$ be a daily availability record for resource r on date d in log L_a, with d starting at time τ_0 and ending at time τ_e. Let P_r be a set of time periods such that $\forall p_i \in P_r : p_i = (\tau_{i,start}, \tau_{i,end}, s_i)$ expresses that resource r has status s during time interval $[\tau_{i,start}, \tau_{i,end}]$, with $[\tau_{i,start}, \tau_{i,end}] \subseteq [\tau_0, \tau_e]$ and $s_i \in \{available, unavailable\} (\forall i \in 1, \dots, n)$. Then, $D_{r,d}$ is represented by $\Upsilon = (r, d, P_r)$, where $\#_k(\Upsilon)$ represents the value of attribute k for daily availability record Υ as suggested for events in Definition 1.

To retrieve daily availability records from an event log, all explicit and implicit information contained in the log is maximally exploited to mark periods during which the resource is either available or unavailable for the process. Daily availability records are retrieved in three steps, i.e. by (i) merging active periods, (ii) identifying boundary pending cases, and (iii) identifying intermediate pending cases [12]. Each of these steps is outlined in the remainder of this subsection. The steps will be exemplified for one specific resource from the running example, which is resource r3. An excerpt from the arrival time imputed activity log for resource r3 is shown in Table 3. For illustrative purposes, the daily availability record only stretches from the start of the first instance in Table 3 until the completion of the last instance. Note that, in reality, $D_{r3,07/04/2018}$ will span the period from 07/04/2018 0:00:00 until 07/04/2018 23:59:59.

(i) Merge active periods. The execution of an activity implies that the resource was certainly available for the process. When multiple activity instances related to the same resource (quasi-)immediately follow each other, they are merged to reflect a period of availability. In Table 3, this holds for the instances associated to cases 104, 82, 86, 107 and 108, 111. The merged availability periods are visualized using dark gray bars in Fig. 3.

The fact that activity instances that quasi-immediately follow each other can also be merged is consistent with the intuition that, e.g., some set-up time is required between handling consecutive cases. To this end, a time tolerance between consecutive activity instances associated to a particular resource can be specified. An appropriate value for this time tolerance is likely to be context-dependent and its specification will rely on domain knowledge. When domain expertise is not present, a log-based recommendation for this time tolerance can be obtained. A simple heuristic builds on the assumption that the set-up time to start a new case depends on the activity duration. If that assumption is valid in a particular context, a percentage of the median activity duration could be used as a time tolerance.

Table 2
Illustration of an arrival time imputed activity log.

Case id	Activity	Resource	$\tau_{arrival}$	τ_{start}	$\tau_{complete}$
...	...	...	...	...	...
72	B	r2	06/04/2018 08:30:04	06/04/2018 08:50:22	06/04/2018 08:56:57
72	C	r3	06/04/2018 08:56:57	06/04/2018 09:02:01	06/04/2018 09:08:42
56	D	r4	06/04/2018 09:00:09	06/04/2018 09:05:51	06/04/2018 09:11:17
38	G	r3	06/04/2018 09:06:22	06/04/2018 09:08:42	06/04/2018 09:11:52
64	D	r4	06/04/2018 08:52:17	06/04/2018 09:11:17	06/04/2018 09:20:41
76	B	r2	06/04/2018 09:02:40	06/04/2018 09:14:21	06/04/2018 09:20:07
64	F	r6	06/04/2018 09:20:41	06/04/2018 09:20:58	06/04/2018 09:24:02
...	...	...	...	...	...

Table 3
Excerpt from arrival time imputed activity log for resource *r3*.

Case id	Activity	Resource	$\tau_{arrival}$	τ_{start}	$\tau_{complete}$
...	...	...	...	...	...
104	C	r3	07/04/2018 11:30:14	07/04/2018 11:32:07	07/04/2018 11:43:27
82	G	r3	07/04/2018 11:38:08	07/04/2018 11:43:27	07/04/2018 11:46:04
86	G	r3	07/04/2018 11:43:52	07/04/2018 12:31:09	07/04/2018 12:33:02
107	C	r3	07/04/2018 12:32:44	07/04/2018 12:33:02	07/04/2018 12:42:46
108	C	r3	07/04/2018 12:51:09	07/04/2018 13:08:17	07/04/2018 13:19:42
111	C	r3	07/04/2018 13:12:47	07/04/2018 13:19:42	07/04/2018 13:25:06
112	C	r3	07/04/2018 13:48:22	07/04/2018 13:48:22	07/04/2018 13:59:22
...	...	...	...	...	...

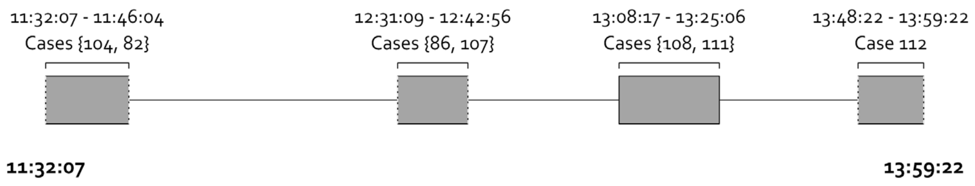


Fig. 3. Merge active periods — illustration.

Definition 4 (*Merged Active Period*)**.** Let $L_a{}^*$ represent an ordered arrival time imputed event log consisting of an ordered set of activity instances η such that $\forall \eta_i, \eta_{i+1} \in L_a{}^* : \#_{\tau_{start}}(\eta_i) \leq \#_{\tau_{start}}(\eta_{i+1})$. Moreover, let $L_{r'}{}^*$ represent the activity instances in $L_a{}^*$ executed by resource r', i.e. $L_{r'}{}^* = \{\eta \in L_a{}^* | \#_r(\eta) = r'\}$.

Then, a merged active period A is an abstraction over a set of activity instances $x \subseteq L_{r'}{}^*$ such that $\forall \eta_i, \eta_{i+1} \in x : (\#_{\tau_{start}}(\eta_{i+1}) - \#_{\tau_{complete}}(\eta_i)) \in [0, \gamma]$, with $\gamma \geq 0$. A merged active period A for resource r is represented by $A = (r, \delta_{start}, \delta_{end})$, with $\delta_{start} = min(\#_{\tau_{start}}(\eta) | \eta \in x\}$ and $\delta_{end} = max(\#_{\tau_{complete}}(\eta) | \eta \in x\}$. $\#_k(A)$ represents the value of attribute k for merged active period A as suggested for events in Definition 1.

(ii) Identify boundary pending cases. While the prior step focused on identifying periods during which the resource was certainly available, the next two steps aim to mark periods of unavailability. A resource is stated to be unavailable during a period a time when cases are present which could be processed by the resource, but no resource activity is recorded. This builds upon the assumption that a resource will work when work items are present and he/she is available for the process. To operationalize this idea, the notion of pending cases is introduced. At a particular point in time, a pending case is a case awaiting service from a resource, i.e. it requested service from a resource, but this request has not yet been fulfilled.

The presence of pending cases is checked at the end of each merged active period first. When so called boundary pending cases, i.e. pending cases at the end of a merged active period, are detected, the resource is unavailable until the start of the following merged active period. Consider, for example, case 86, which requested service from resource r3 at 11:43:52. Hence, case 86 was awaiting service from resource r3 at the end of this resource's first active period, but no resource activity is recorded until 12:31:09. As a consequence, r3 is marked unavailable between these active periods. This is shown in Fig. 4 using a dashed (light gray) box.

When multiple resources can execute a particular activity and a single queue is present, even more insights can be retrieved from the event log. A single queue implies that several resources share a queue for a particular activity, e.g. one waiting room with multiple counters to serve cases. Under these circumstances (multiple resources and a single queue), pending cases which are eventually served by another resource also qualify as boundary pending cases for the resource under analysis. This is due to the fact that the resource under analysis could have processed these cases, but no resource activity has been registered. Consider, for instance, that a single queue prevails at activity G, which can be executed by r3 or r7. Furthermore, suppose that a pending case is present at the end of a merged active period for r3 which is eventually handled by r7. The presence of this pending case and the absence of recorded activity for r3 shows that r3 is unavailable for a particular period of time. This unavailable period for r3 starts at the end of the merged active period and stops when r7 starts processing the pending case. However, when r7 starts handling the pending case after the start of the next merged active period of r3, the unavailable period of r3 ends when this next merged active period starts.

To formalize boundary pending cases, the generic concept of a pending case is first defined. Afterwards, the notion of a boundary pending case is specified.

Definition 5 (*Pending Case*)**.** Let α be the set of activity labels in $L_a{}^*$ and let ρ be the set of resources in $L_a{}^*$. Moreover, let $\alpha_{r'}$ represent the set of activity labels that resource r' can execute, i.e. $\alpha_{r'} = \{a' \subseteq \alpha | \exists \eta \in L_a{}^* : \#_a(\eta) = a' \wedge \#_r(\eta) = r'\}$. Then, a pending case for resource r' at time t is an activity instance $\eta \in L_a$ for which the following conditions cumulatively hold:

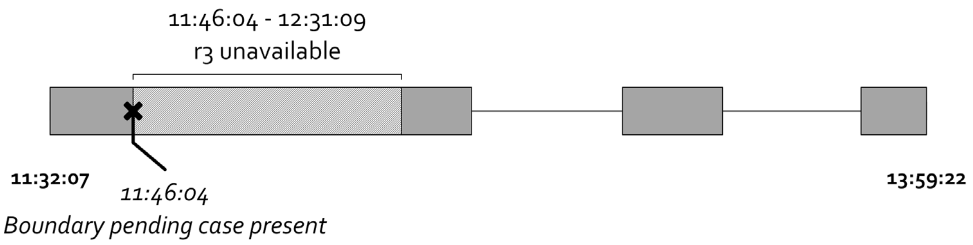


Fig. 4. Identify boundary pending cases — illustration.

1. $\nexists \eta \in L_{r'}{}^* : t \in [\#_{\tau_{start}}(\eta), \#_{\tau_{complete}}(\eta)]$, i.e. resource r' is not active at time t
2. **One of** the following conditions holds:
   - $\#_{\tau_{arrival}}(\eta) \leq t \wedge \#_{\tau_{start}}(\eta) > t \wedge \#_r(\eta) = r'$
   - $\#_{\tau_{arrival}}(\eta) \leq t \wedge \#_{\tau_{start}}(\eta) > t \wedge \#_r(\eta) = \rho \setminus r' \wedge \#_a(\eta) \in \alpha_{r'}$

Definition 6 (*Boundary Pending Case*). Let $A_{r'}$ be a merged active period for resource r' (cf. Definition 4). Then, a boundary pending case is an activity instance $\eta \in L_a{}^*$ for which one of the conditions in Definition 5 holds at time $t = \#_{\delta_{end}}(A_{r'})$.

(iii) Identify intermediate pending cases. When no boundary pending cases are present, the first resource request issued after the end of a particular merged active period is considered. When this request is not fulfilled immediately while no resource activity is recorded, the respective case is called an intermediate pending case. When such a case is present, the resource is marked unavailable from the time the request is issued until the start of the next merged active period. For example: from Table 3, it follows that case 108 requested r3 at 12:51:09, while the next active period of r3 started at 13:08:17. Hence, an unavailable period is introduced, as shown in Fig. 5.

Similar to boundary pending cases, intermediate pending cases can also originate from resource requests which are eventually fulfilled by another resource when an activity has a single queue. Consider, once again, activity G, which can be carried out by either r3 or r7. Suppose that case 93 requests service for activity G at 13:28:08 and r7 fulfills this request at 13:42:51 by starting to process this case. As no resource activity is recorded for r3 in that period of time, r3 is unavailable within this time window, as shown in Fig. 6.

Definition 7 (*Intermediate Pending Case*). Let $\mathscr{A}_{r'}$ be the set of merged active periods for resource r' (cf. Definition 4). Moreover, let $A_{r',1}$ and $A_{r',2}$ be two subsequent merged active periods for r', i.e. $\#_{\delta_{start}}(A_{r',2}) > \#_{\delta_{end}}(A_{r',1}) \wedge \nexists A_{r',x} \in \mathscr{A}_{r'} : \#_{\delta_{end}}(A_{r',1}) \leq \#_{\delta_{start}}(A_{r',x}) \leq \#_{\delta_{start}}(A_{r',2})$.

Then, an intermediate pending case is an activity instance $\eta \in L_a{}^*$ for which one of the conditions in Definition 5 holds for one of the times $t \in \{\#_{\tau_{arrival}}(\eta) | \#_{\delta_{end}}(A_{r',1}) < \#_{\tau_{arrival}}(\eta) < \#_{\delta_{start}}(A_{r',2})\}$.

Outcome. The previous steps extract all explicit and implicit information from the log to identify periods during which a resource is available or unavailable for the process. The remaining time periods, marked with a question mark in Fig. 7, are unassigned periods. The latter are time frames during which no resource activity is recorded and no work is present for the resource under consideration. For unassigned periods, the log does not provide any evidence to assign these periods as either available or unavailable.

3.2.3. Daily availability record post-processing

Contingent upon process characteristics such as the typical workload and the typical processing time, the obtained daily availability records can be highly fragmented. When high fragmentation prevails, post-processing efforts can increase the comprehensibility of the daily availability records. Moreover, unassigned periods can be removed during post-processing, which is required to use daily availability records to build resource availability calendars. In Martin et al. [12], three potential post-processing actions are suggested: (i) working day specification, (ii) unassigned period imputation and (iii) unavailability period threshold enforcement.

(i) Working day specification. A first post-processing act involves the specification of the start and end of a resource's working day. These can, consistent with Wombacher et al. [40], be proxied by the start of the first activity instance of a resource on a particular day and the completion of the last instance of this resource on that day, respectively. Integrating the working day notion simplifies the daily availability records as a resource is considered to be unavailable for the process outside the working day. Moreover, a resource is assumed to be unavailable for the process on days where no resource activity is recorded.

(ii) Unassigned period imputation. During post-processing, unassigned periods (visualized using a question mark in Fig. 7) can be assigned as an available period, an unavailable period or a mixture of both. This reduces daily availability record fragmentation and seems warranted, especially when the unassigned periods are relatively short. As an appropriate imputation mechanism can be context-dependent, discussions with domain experts on this matter are warranted. Alternatively, a simple imputation heuristic could designate unassigned periods based on the type of period before and after an unassigned period. When both the prior and the next period are of the same type, i.e. both available or both unavailable, the unassigned period is also assumed to be of that type. In contrast, when the resource is available in the prior period and unavailable in the next period (or vice versa), the unassigned period is split proportionate to the relative length of the adjacent periods.

(iii) Unavailability period threshold enforcement. Another potential post-processing step implies the enforcement of an unavailability period threshold. Only unavailability periods that are longer than this boundary value will be retained. Specifying such a threshold is suitable as unavailability is defined in a rigid way when retrieving daily availability records. During the creation of daily availability records, a resource is considered to be unavailable when he/she can potentially work on a case (i.e. a pending case), but refrains from doing so. However, this does not take into account behavior such as deferring particular cases to a more experienced colleague. Introducing a threshold can both reduce the fragmentation of the daily availability records and better aligns the remaining unavailable periods with their conceptual meaning, i.e. periods capturing breaks or the resource's assignment to another process. Determining an appropriate boundary value will be context-dependent and, hence, requires domain knowledge.

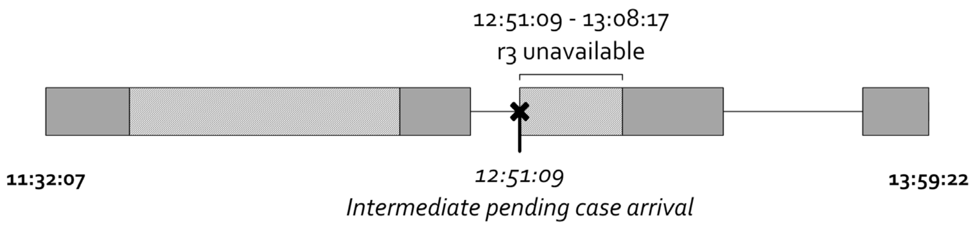


Fig. 5. Identify intermediate pending cases — illustration of request fulfilled by resource under analysis.

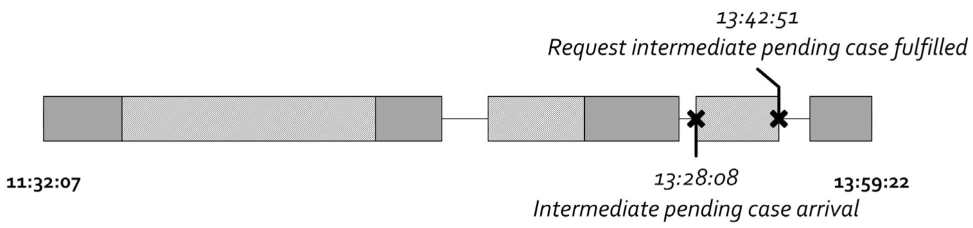

13:42:51
Request intermediate pending case fulfilled
11:32:07
13:28:08
Intermediate pending case arrival
13:59:22

Fig. 6. Identify intermediate pending cases — illustration of request fulfilled by another resource in a single queue situation.

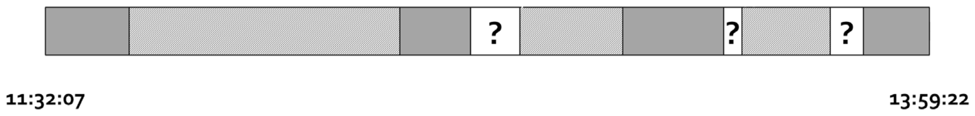


Fig. 7. Daily availability record — illustration.

3.3. Stage 2: Resource availability calendar creation

Stage 1 of the method generates a daily availability record for each resource for each day. While the daily availability records can already be useful for performance management purposes, the definition of resource availability calendars as input for, e.g., a process simulation model requires additional efforts. When building a simulation model, the specification of resource availability is an important modeling task [35]. Given its implications on the flow of cases through the process, it is important that the availability calendar takes into account the temporal dimension of availability, i.e. at which time of day a resource is available, and intermediate availability interruptions due to, e.g., a break. As these are the key characteristics of the mined daily availability records, they form an excellent building block when defining resource availability calendars having an empirical basis.

To operationalize the use of daily availability records to determine resource availability calendars, two sampling approaches are presented: (i) direct sampling, and (ii) cluster-based sampling. Following a sampling approach implies that the daily availability records will directly be used in the resource availability calendars. An alternative approach could involve aggregating daily availability records to create an "average" availability calendar in terms of, amongst others, the start of the working day, the length of the working day, and the number of intermediate availability interruptions. However, aggregation entails the risk that the resulting availability calendar does not reflect actual availability patterns with a sufficient degree of accuracy. From this perspective, a sampling approach is deemed the most suitable as the daily availability records, which reflect real-life availability patterns from data, are directly used.

Both sampling approaches require the selection of a date classification mechanism $\mathscr{T}$ that is suitable for the application at hand. The date classification mechanism will subdivide the daily availability records for a particular resource in groups, which will be used in the sampling approach, as will be outlined later in this subsection. Examples include $\mathscr{T} = \{Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday\}$ and $\mathscr{T} = \{weekday, weekend\}$. However, the date classification mechanism can also be more complex. Consider, for instance, that resource availability varies over the days of the week, but that seasonal effects cause resource availability to be significantly higher during summer than during winter. Under these circumstances, a distinction between a particular weekday during summer months and during winter months can be introduced. Consequently, $\mathscr{T}$ would be of the form $\mathscr{T} = \{Monday_s, Tuesday_s, Wednesday_s, Thursday_s, Friday_s, Saturday_s, Sunday_s, Monday_w, Tuesday_w, Wednesday_w, Thursday_w, Friday_w, Saturday_w, Sunday_w\}$, where s and w refer to summer and winter, respectively. This illustrates that domain knowledge is required to specify a suitable date classification mechanism as it can be highly context-specific. However, this does not undermine the usefulness of daily availability records for the specification of resource availability calendars. While defining the date classification mechanism, it should be taken into account that sufficient data is available to back it. For instance: when event logs are only available for summer months, no daily availability records will be available for the winter months.

Definition 8 (*Resource Availability Calendar*)**.** Let $S_{r'}$ represent the resource availability calendar for resource r', consisting of a set of daily availability records. Let $\mathscr{D}$ be the set of daily availability records defined over log $L_a{}^*$ and let $\mathscr{D}_{r'}$ be the set of daily availability records for resource r', i.e. $\mathscr{D}_{r'} = \{\Upsilon \in \mathscr{D} \,|\, \#_r(\Upsilon) = r'\}$. Let $\mathscr{T}$ be the date classification mechanism and let function f be a mapping $f : d \to t$, where d represents a date and $t \in \mathscr{T}$. When $S_{r'}$ is defined over date-time interval $[\tau_{s,start}, \tau_{s,end}]$, then the entry in $S_{r'}$ for date d is determined by applying sampling method σ on $\mathscr{D}^{\star}_{r'} = \{\Upsilon \in \mathscr{D}_{r'} \,|\, f(\Upsilon) = f(d)\}$. This is repeated $\forall d \in [\tau_{s,start}, \tau_{s,end}]$.

Fig. 8 illustrates the structure of a one-week resource availability calendar for resource r3 with $\mathscr{T} = \{Monday, Tuesday, Wednesday, Thursday, Friday\}$. It shows, for instance, that the availability of this resource for the process is rather limited on Tuesday and Friday.

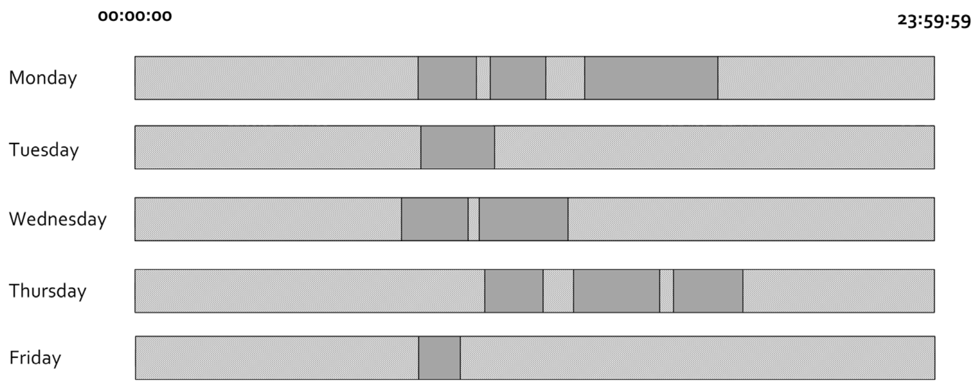


Fig. 8. Resource availability calendar — illustration.

The remainder of this subsection proposes the two sampling methods σ, i.e. direct sampling and cluster-based sampling.

3.3.1. *Direct sampling*

The direct sampling method to create resource availability calendars uses random sampling. Suppose that a two-week availability calendar is required, e.g. as input for a process simulation model, and that the date classification mechanism $\mathscr{T} = \{Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday\}$ suits the application context. In that case, all daily availability records for that resource are grouped based on the day of the week they represent. For instance: all daily availability records for dates that are Mondays are grouped. When a Monday is required in the resource availability calendar, a daily availability record is randomly sampled from this daily availability record subset. Suppose that resource r3 typically works full days on Monday, with a lunch break at noon, but occasionally continues to work during his lunch break to leave in the afternoon. The fact that the latter availability pattern is less frequent will be reflected by the number of daily availability records having this structure. Consequently, the probability that it is drawn using random sampling is lower. In this way, the direct sampling approach makes sure that frequently recurring availability patterns are included in the availability calendar more often than occasional patterns.

3.3.2. *Cluster-based sampling*

With direct sampling, each daily availability record can potentially be selected to compose a resource's availability calendar. In cluster-based sampling, a limited number of "representative" daily availability records are identified to sample from when composing a resource availability calendar.

Cluster analysis involves grouping objects in clusters such that clustered objects are more similar to each other than to objects in other clusters. Several procedures to perform clustering have been proposed in literature. In general, a distinction can be made between hierarchical and non-hierarchical clustering. Hierarchical clustering develops a treelike structure in which clusters (initially consisting of a single object) are iteratively combined into larger clusters. In this way, the resulting treelike structure provides an overview of all possible cluster solutions. In contrast, non-hierarchical clustering generates a single solution given the number of clusters defined by the user. Despite the fact that only a single cluster solution is generated, non-hierarchical clustering is often applied as it, e.g., tends to be less influenced by outliers and tends to be more scalable for larger datasets. Hierarchical and non-hierarchical clustering can also be combined: hierarchical clustering is used first to identify the appropriate number of clusters, after which non-hierarchical clustering groups objects in this number of clusters [48]. For a more elaborate overview on clustering, the reader is referred to textbooks such as Hair et al. [48] and Witten et al. [49].

In this paper, the objects that are clustered are daily availability records. Cluster-based sampling also uses a date classification mechanism to group daily availability records. However, instead of randomly selecting a daily availability record from an appropriate group (which holds for direct sampling), a cluster analysis is performed for this group. Consider, once again, that the date classification mechanism $\mathscr{T} = \{Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday\}$ applies. For each resource, daily availability records are grouped for each day of the week, which is followed by a separate cluster analysis for each group.

Even though a discussion of the implementation details will follow in Section 3.4, it should be noted that a two-stage clustering procedure is followed. Hierarchical clustering is used first to determine the appropriate number of clusters, after which partitioning around medoids is used to obtain a cluster solution for this number of clusters. Partitioning around medoids operates similar to the well-known k-means clustering technique [50]. However, in contrast to k-means, a cluster is characterized by a medoid and not by its mean or centroid. While a mean or a centroid does not need to correspond to a cluster element, a medoid is always a cluster element, i.e. a specific daily availability record [51]. More specifically, it is the cluster element for which the mean similarity to all other cluster elements is maximal. To determine the cluster solution using partitioning around medoids, an initial set of medoids is randomly selected and the other objects are assigned to their nearest medoid. To this end, a distance measure needs to be specified, which will be proposed when the implementation is discussed. After the initial assignment, a swapping strategy is applied to determine whether the mean similarity within a cluster can be increased when another cluster element becomes the medoid. If this is the case, the medoids change and it is checked whether cluster elements need to be reassigned to another cluster. Medoid swapping and cluster element reallocation is repeated until the location of the medoids stabilizes [50,51].

For the purpose of resource availability calendar creation, the medoids of the final cluster solution will be considered "representative" for their respective cluster. Consequently, during availability calendar composition, only cluster medoids can be selected. The size of each cluster will determine the probability with which its centroid will be selected.

Fig. 9 visualizes the difference between direct sampling and cluster-based sampling. Each square in this figure represents a daily availability record for a particular date classification mechanism element for a particular resource, e.g. each square could represent a Monday of resource r3. When direct sampling is used (Fig. 9a), one of the daily availability records is randomly selected for each Monday in the availability calendar. When cluster-based sampling is applied (Fig. 9b), clusters are first formed.

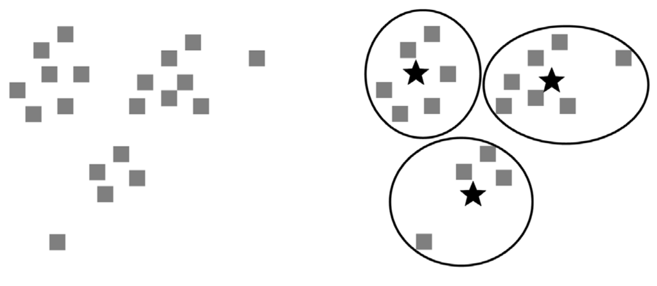


Fig. 9. Illustration of sampling methods.

The medoids, visualized using star shapes, are the only daily availability records that can be selected. The selection probability is determined by the cluster size.

From the previous, it follows that cluster-based sampling is less sensitive to outliers as medoids are used for availability calendar construction. However, compared to direct sampling, more data should be available as a cluster analysis is performed for each date classification mechanism element for each resource. Even though generic rules of thumb to determine a minimal cluster analysis sample size are absent [52], the number of daily availability records for each date classification mechanism element should be sufficiently large to adequately represent the variety of availability patterns for a particular resource [48]. As a cluster analysis is performed for each date classification mechanism separately, the required event log size will also depend on the selected date classification mechanism. The more granular it is, the longer the event log's timespan will need to be to obtain a valuable cluster solution.

3.4. Implementation

The resource availability calendar identification method is fully implemented[2] using R,[3] which is a programming language providing extensive functionalities for data manipulation and statistical analysis. The key packages that are used are `dplyr` for data manipulations and summarizations, `lubridate` to work with timestamps and `reshape` for converting the event log to an activity log.

The implementation encompasses both stages of the proposed method, i.e. the retrieval of daily availability records and the creation of resource availability calendars. Appendix contains the pseudocode for three key algorithms: (i) daily availability record retrieval, (ii) resource availability calendar creation using direct sampling, and (iii) resource availability calendar creation using cluster-based sampling.

The cluster-based sampling method for resource availability calendar creation is implemented using a two-stage clustering method, as suggested in Hair et al. [48]. This implies that the number of clusters is determined using hierarchical clustering, after which a cluster solution with this number of clusters is generated using non-hierarchical clustering. Both hierarchical and non-hierarchical clustering require a distance measure to express the similarity between objects, in this case between daily availability records. The implementation provides the functionality to specify the distance between two daily availability records as a weighted average of:

- the degree of non-correspondence, which quantifies the total time during which the period type (i.e. available or unavailable) differs between both daily availability records
- the total available time over the course of the day

Altering the weight of both variables changes the way in which the distance measure that guides the clustering process is calculated. However, the fact that two variables are provided in the implementation does not limit the potential for further customization. A user can further customize the similarity measure to make it suitable in a particular application context by, e.g., adding additional variables.

To determine the number of clusters that will be used for non-hierarchical clustering, hierarchical clustering as implemented in the `stats` package is used. In the hierarchical clustering process, average linkage is used to determine the similarity between clusters, implying that similarity is calculated taking into account all cluster elements instead of, e.g., a single one in single linkage [48]. To automatically retrieve an appropriate amount of clusters from the hierarchical clustering output, the silhouette method implemented in the `factoextra` package is used. The silhouette method determines the average silhouette for different numbers of clusters, which is based on the distance between an object and (i) other objects of the same cluster and (ii) objects in other clusters. The number of clusters for which the highest average silhouette is obtained is selected [53]. Given this number of clusters, non-hierarchical clustering is used to create clusters of daily availability records. To this end, partitioning around medoids as implemented in the `cluster` package is used.

4. Evaluation

To evaluate the proposed method, a distinction is made between the two stages shown in Fig. 2. The first stage, i.e. the accuracy of the daily availability records, is evaluated in Section 4.1 by determining the degree of non-correspondence between the real calendar and the mined daily availability records. In Section 4.2, the two sampling methods introduced to operationalize the second stage are evaluated by comparing simulation outputs using the real calendar with their corresponding values when one of the sampled resource availability calendars is used. Explicit research questions will be identified for each experiment [54].

4.1. Daily availability record accuracy

The daily availability records are the key building blocks for the construction of resource availability calendars. To this end, this experiment aims to evaluate the accuracy of the daily availability records retrieved from an event log (stage 1 of the method). Hence, the main research question is: *"How accurate are daily availability records retrieved from event logs as a representation of the real availability of a resource?"*. Synthetic event logs are used for empirical testing as this enables a performance analysis in a controlled environment. The experimental design is outlined in Section 4.1.1, the results are presented in Section 4.1.2.

4.1.1. Experimental design

To evaluate the accuracy of the daily availability records, a synthetic event log is created based on the running example introduced in Section 2.3. For each of the seven resources, a separate day calendar is generated, representing the periods of time during which the resource is available for the process. This day calendar is repeated each day in the generated event log. Note that the used process model is kept fairly simple from the control-flow perspective as control-flow complexity will not influence the method's ability to retrieve resource availability calendars. The method focuses on resources and activities, and not on the order

[2] The full implementation is available at https://github.com/nielsmartin/resource_availability_calendars.

[3] https://www.r-project.org.

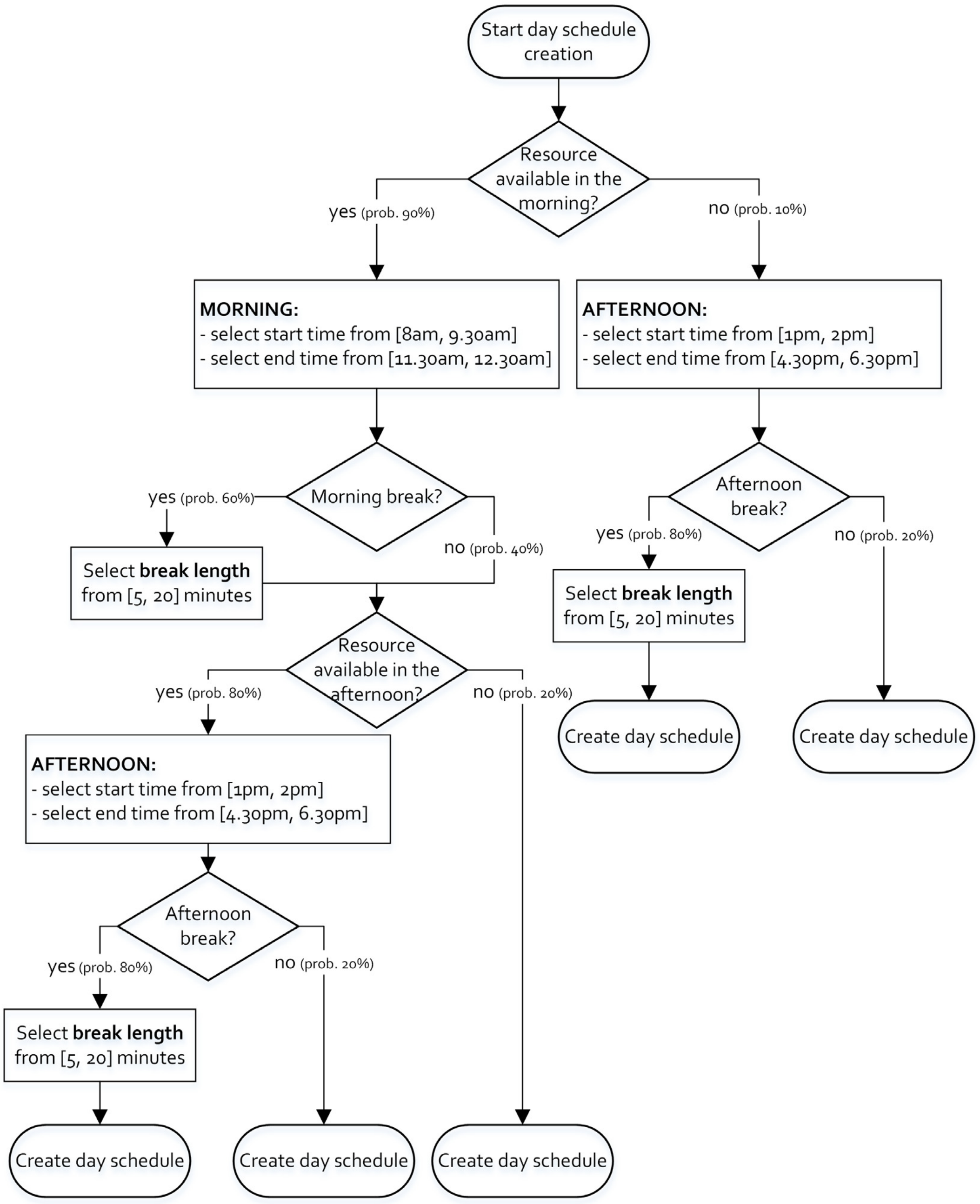

Start day schedule creation
Resource available in the morning?
yes (prob. 90%)
no (prob. 10%)
MORNING:
- select start time from [8am, 9.30am]
- select end time from [11.30am, 12.30am]
AFTERNOON:
- select start time from [1pm, 2pm]
- select end time from [4.30pm, 6.30pm]
Morning break?
yes (prob. 60%)
no (prob. 40%)
Select break length from [5, 20] minutes
Afternoon break?
yes (prob. 80%)
no (prob. 20%)
Select break length from [5, 20] minutes
Create day schedule
Create day schedule
Resource available in the afternoon?
yes (prob. 80%)
no (prob. 20%)
AFTERNOON:
- select start time from [1pm, 2pm]
- select end time from [4.30pm, 6.30pm]
Afternoon break?
yes (prob. 80%)
no (prob. 20%)
Select break length from [5, 20] minutes
Create day schedule
Create day schedule
Create day schedule

Fig. 10. Day calendar creation procedure in synthetic event logs.

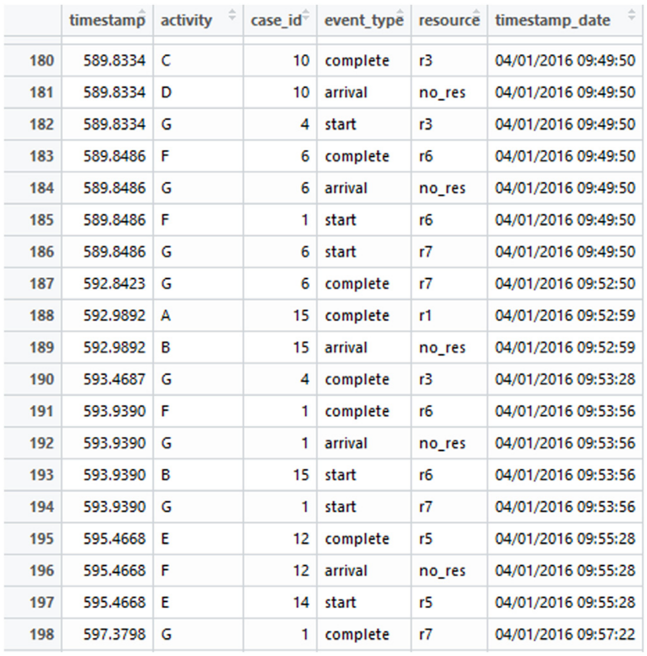
of activities. Only when the arrival time of a case at an activity needs to be imputed, as discussed in Section 3.2.1, a control-flow notion such as the prior activity might be required.

The procedure used to create the aforementioned day calendar consists of a series of choices that the synthetic data generator autonomously makes. As shown in the decision flowchart in Fig. 10, a day calendar can be generated in which the resource is only available in the morning or both in the morning and in the afternoon. Moreover, the availability period in each of these parts of day can be interrupted by a break. Following this procedure ensures that calendars with varying structures will be generated and used for the creation of synthetic event logs. Variability is introduced in terms of the length of availability periods, the number and length of intermediate interruptions such as breaks, and whether the resource is available in both the morning and the afternoon. Note that the knowledge on the generated calendars will only be used for evaluation purposes and is not passed in any way to the daily availability record retrieval method.

Using the created calendars and the parameters annotated in Fig. 1, a synthetic event log containing process execution information for four weeks is generated. Case interarrival times and activity durations are randomly drawn from the probability distributions specified in Fig. 1. From a technical perspective, the synthetic event log is created using `SimPy`,[4] which is an open-source framework for discrete-event simulation in the programming language `Python`.[5] To illustrate the structure of the synthetic event logs, a screenshot of such a log is included in Fig. 11. Events represent the arrival of a case at an activity, the

4 https://simpy.readthedocs.io.

5 The source code of the simulator is available at https://github.com/nielsmartin/resource_availability_calendars.



	timestamp	activity	case_id	event_type	resource	timestamp_date
180	589.8334	C	10	complete	r3	04/01/2016 09:49:50
181	589.8334	D	10	arrival	no_res	04/01/2016 09:49:50
182	589.8334	G	4	start	r3	04/01/2016 09:49:50
183	589.8486	F	6	complete	r6	04/01/2016 09:49:50
184	589.8486	G	6	arrival	no_res	04/01/2016 09:49:50
185	589.8486	F	1	start	r6	04/01/2016 09:49:50
186	589.8486	G	6	start	r7	04/01/2016 09:49:50
187	592.8423	G	6	complete	r7	04/01/2016 09:52:50
188	592.9892	A	15	complete	r1	04/01/2016 09:52:59
189	592.9892	B	15	arrival	no_res	04/01/2016 09:52:59
190	593.4687	G	4	complete	r3	04/01/2016 09:53:28
191	593.9390	F	1	complete	r6	04/01/2016 09:53:56
192	593.9390	G	1	arrival	no_res	04/01/2016 09:53:56
193	593.9390	B	15	start	r6	04/01/2016 09:53:56
194	593.9390	G	1	start	r7	04/01/2016 09:53:56
195	595.4668	E	12	complete	r5	04/01/2016 09:55:28
196	595.4668	F	12	arrival	no_res	04/01/2016 09:55:28
197	595.4668	E	14	start	r5	04/01/2016 09:55:28
198	597.3798	G	1	complete	r7	04/01/2016 09:57:22

Fig. 11. Screenshot of a synthetic event log.

start of activity execution, or the completion of activity execution. Note that the arrival events in Fig. 11 build upon the assumption that a case arrives at an activity immediately after the prior activity has been completed. When the system does not record arrival events, which is not required according to Definition 1, they can be imputed as discussed in Section 3.2.1.

Solely using the event log, the method is executed with a time tolerance for active period merging of zero (i.e. active periods are only merged when they immediately follow each other) and single queues prevailing (i.e. when multiple resources can execute an activity, they share the same queue). Moreover, the first two post-processing steps, outlined in Section 3.2.3, are applied to the generated daily availability records: (i) the notion of a working day is introduced, and (ii) the unassigned periods are imputed using the outlined heuristic. The latter enables a one-on-one comparison between the imputed daily availability records and the real calendars as both are only composed of available and unavailable periods. An unavailability period threshold, which constitutes the third post-processing step, is not defined. This makes the evaluation more strict as short unavailability periods that do not represent genuine interruptions of availability will reduce the daily availability records' accuracy.

To quantify the accuracy of the retrieved daily availability records, they are compared to the real calendars used to generate the synthetic event log. To this end, the degree of non-correspondence is determined between the real calendar for a particular day and resource on the one hand and the daily availability record for that same day and resource on the other hand. A non-corresponding period of time is defined as a time frame in which the daily availability record predicts another period type than the one prevailing in reality, with the period type being either available or unavailable. To place the non-corresponding time into perspective, it is expressed relative to the length of the working day according to the real calendar. The latter corresponds to the time difference between the start of the first and the end of the last activity instance associated to a particular resource on that day.

To thoroughly evaluate the accuracy of daily availability record retrieval, the procedure outlined above is repeated 100 times, i.e. 100 synthetic event logs with different underlying calendars are created. For each of these logs, the retrieval method is applied and the degree of non-correspondence is calculated for each daily availability record.

Table 4
Summary statistics of non-correspondence on the resource level (percentage values).

Resource	Mean	sd	Median	min	q1	q3	max
r1	0.49	1.93	0.00	0.00	0.00	0.00	15.29
r2	29.77	18.08	25.96	0.00	16.67	45.18	79.01
r3	10.80	14.89	0.00	0.00	0.00	19.26	60.81
r4	14.34	17.45	5.21	0.00	0.00	25.01	71.39
r5	33.28	21.93	30.07	0.00	16.73	47.70	96.15
r6	7.55	12.93	0.00	0.00	0.00	10.34	51.72
r7	47.63	15.41	46.39	0.00	37.65	56.42	99.04

4.1.2. Results

Over all 100 synthetic event logs, the mean degree of non-correspondence equals 20.54%, with an associated standard deviation of 22.12%. The median value is 14.35% and the first and third quartile are equal to 0% and 37.37%, respectively. This implies that, on average, the daily availability records correctly rediscover the available/unavailable status of the resource for 79.46% of the timespan of a working day.

To gain a more detailed insight in the performance of the daily availability records retrieval method, Table 4 presents the non-correspondence summary statistics for each resource. From this table, it follows that the degree of non-correspondence strongly varies depending on the resource under consideration. The mean values range from 0.49% for r1 to 47.63% for r7.

To illustrate the variation in rediscovery performance, two daily availability records are compared to their corresponding real calendars. Fig. 12 presents a real calendar for resource r4, consisting of both a morning and afternoon availability period. Both of them are interrupted for a break. This real calendar is perfectly rediscovered, i.e. the non-correspondence equals zero. In contrast, the daily availability record with the highest non-correspondence value (99.04%), which is associated to r7, is visualized in Fig. 13. While the resource is, in reality, available in [09:00:00, 11:32:00], the daily availability record indicates that the resource is only available in [09:06:00, 09:07:24]. This is due to the fact that r7 only executes a single activity on this day, starting at 09:06:00 and ending at 09:07:24. Consequently, when applying the working day post-processing step, the working day is limited to this time frame in the daily availability record and the resource is considered to be unavailable outside this period.

From the illustration in Fig. 13, it follows that an important explanation for the high non-correspondence detected for some daily availability records is the false classification of idle periods as unavailable because they are outside the daily availability record's working day. The latter tends to occur on days at which limited resource activity is recorded as the boundaries of the daily availability record's working day are the start of the first activity instance and the completion of the last one for a resource on a particular day. This intuition is confirmed by Table 5, which summarizes the working day length according to the real calendar and the daily availability records. Moreover, the ratio between both is reported. From this ratio, it follows that resources with higher non-correspondence values tend to have shorter working days in the daily availability records than the ones prevailing in reality. For instance: for r7, the resource with the highest mean non-correspondence, the working days in the daily availability records only span, on average, 75.77% of the real working days.

The observation that resources are sometimes available, but not performing any work, as is the case in Fig. 13, indicates

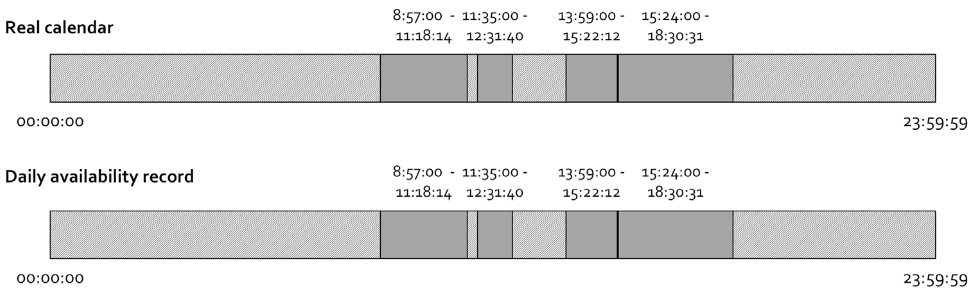


Fig. 12. Degree of non-correspondence — example of perfect rediscovery.

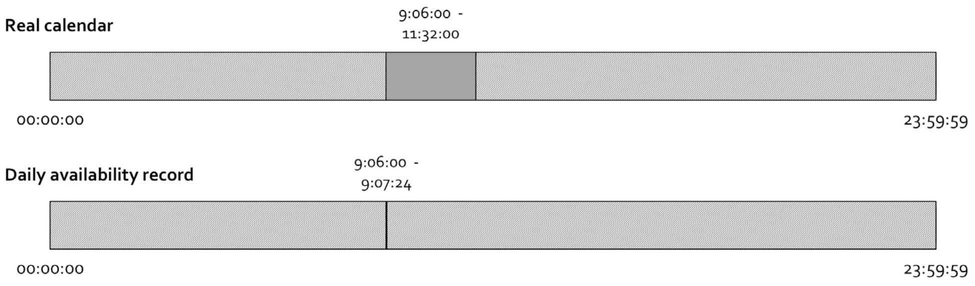


Fig. 13. Degree of non-correspondence — example of high non-correspondence.

Table 5
Mean working day length according to real calendar and daily availability records.

Resource	Working day length[a]		Ratio[b]
	Real calendar	Daily avail. rec.	
r1	401.96	399.33	99.34
r2	461.63	360.44	78.08
r3	435.37	391.27	89.87
r4	475.25	416.86	87.82
r5	431.47	327.54	75.91
r6	422.08	389.17	92.20
r7	414.14	313.80	75.77

[a]Expressed in minutes.
[b]Expressed as percentage values.

Table 6
Summary statistics of resource utilization on the resource level (percentage values).

Resource	Mean	sd	Median	min	q1	q3	max
r1	98.75	4.04	100.00	68.43	100.00	100.00	100.00
r2	51.33	22.31	51.53	10.31	34.13	64.38	100.00
r3	82.21	21.78	97.51	25.34	63.15	100.00	100.00
r4	78.45	22.12	84.84	15.74	59.90	100.00	100.00
r5	47.15	23.37	45.58	2.74	29.63	60.39	100.00
r6	87.24	19.02	100.00	28.30	77.27	100.00	100.00
r7	24.44	13.65	21.71	0.66	14.87	31.97	92.47

that resources are idle. To investigate the degree of idleness, the resource utilization is calculated, which expresses the percentage of the available time that a resource is actually performing work. Summary statistics on the resource utilization are provided for each resource in Table 6. As was the case for non-correspondence values, important differences related to resource utilization are observed among resources. For instance: mean values range between 24.44% for r7 and 98.75% for r1. Resource utilization tends to be higher for resources that are active in the upstream segment of the process, such as r1 in activity A and r6 in activity B. Downstream resources rely on the input from upstream resources and have a tendency to be idle more often. An example of the latter is r7, which is only responsible for the execution of the last activity in the process. Moreover, downstream resources will be idle when process execution starts as the process is 'empty' at the start. When resources are ordered based on increasing mean non-correspondence and decreasing mean utilization, the same order is obtained, i.e. $r1 \rightarrow r6 \rightarrow r3 \rightarrow r4 \rightarrow r2 \rightarrow r5 \rightarrow r7$. A similar relationship holds for the median values when ranking ties are disregarded. This indicates that resource utilization influences the accuracy of the retrieved daily availability records.

4.2. Sampling method evaluation

To generate resource availability calendars from daily availability records (stage 2 of the method), two sampling methods are introduced: direct sampling and cluster-based sampling. To evaluate these methods, the effect of using the generated calendars as an input for a process simulation model is determined. Consequently, the main research question is *"How closely do the outputs of simulation models in which resource availability is modeled using sampling methods resemble the outputs of a model containing the real availability?"*. To this end, simulation outputs using the real calendar are compared to their corresponding values when one of the generated calendars is used. The experimental design is outlined in more detail in Section 4.2.1 and some key results are discussed in Section 4.2.2.

4.2.1. Experimental design

To evaluate the proposed sampling methods, a synthetic event log is generated based on the running example shown in Fig. 1. The event log contains process execution information for a period of 12 weeks. For each resource, a distinct calendar is randomly generated. This calendar consists of a unique availability pattern for each day, which is generated following a similar procedure as the one shown in Fig. 10. The only difference is that it is determined a priori whether a resource is available full time for the process or only half time. While resources r5 and r7 are available half time, the other resources are available full time. For resources which are only available half time, it is randomly determined whether they are available to the process in the morning or in the afternoon.

Solely using the event log as an input, daily availability records are retrieved with a time tolerance for active period merging of 10% of the median activity duration (to account for, e.g., set-up time) and single queues prevailing. Moreover, the following post-processing steps are applied: working day specification and unassigned period imputation. The generated daily availability records are the starting point for resource availability calendar creation. Both direct sampling and cluster-based sampling are applied to retrieve an availability calendar for each resource.

For both types generated availability calendars, a simulation model is created using the respective availability calendar as an input. This model is simulated for six weeks and three process performance measures are calculated: the case's cycle time, the number of activity instances per resource, and the number of entities leaving the process after being fully handled. For comparison purposes, a simulation model with the real calendar is also created and simulated for six weeks. To ensure comparability, common random numbers are used for the entity arrival rate, activity durations and the logic at the decision point following activity B in all simulation models. This ensures that differences in simulation outputs can solely be attributed to differences between the real calendar and the resource availability calendar created by the method.

To account for the randomness in the resource availability calendar development method, the creation of resource availability calendars and the determination of process performance measures is repeated 20 times.

4.2.2. Results

The creation of resource availability calendars solely using an event log as an input is repeated 20 times. Each time, the created availability calendars are included in a simulation model and this model is executed. The (absolute value of the) percentage deviations between the obtained simulation outputs and their equivalents using the real calendars are calculated. Table 7 summarizes the results for the aforementioned process performance measures. The second column contains the output values for the simulation with the real calendar. The third and fourth column provide summary statistics (mean/standard deviation/minimum/maximum) on the percentage deviation over all 20 simulations between outputs with the real calendar and (i) a direct sampling availability calendar and (ii) a cluster-based sampling availability calendar, respectively.

From Table 7, it follows that including data-driven resource availability calendars created using the introduced method generates simulation results which closely resemble their equivalents using the real calendar. For example: the mean deviation for the average cycle time equals 3.90% when direct sampling is applied and 4.84% when cluster-based sampling is used. The largest deviation is observed for the number of activity instances per resource for r7, with a mean deviation of 10.85% when using a direct sampling availability calendar and 14.43% with a cluster-based sampling availability calendar. It should be noted that, for r7, simulations with data-driven availability calendars generated lower output values than their equivalents using the real calendar. This can be attributed to the fact that, for r7, some daily availability records can underestimate the real availability. Reasons for this observation have been discussed in Section 4.1.2.

For the generated event log, simulation outputs using direct sampling resemble reality more closely than cluster-based sampling. This can be attributed to the fact that the randomly composed calendar to generate the event log serving as input for the method is fairly stable and does not contain outliers. As direct sampling allows for a wider range of daily availability records to be selected compared to cluster-based sampling, it can capture the subtle variations present in the real calendar slightly better. However, this does not imply that direct sampling is always preferable. Consider, for instance, a situation in which the event log contains an important number of availability patterns which are not representative (i.e. outliers). Under such circumstances, cluster-based sampling is likely to be more suitable as only cluster medoids will be taken into consideration during calendar creation.

5. Discussion

The discussion is centered around three topics of the proposed method: (i) the applicability, (ii) the generalizability, and (iii) the limitations.

5.1. Applicability

The method proposed in this paper retrieves resource availability calendars from an event log. The method solely takes data as a starting point and does not rely on a process model. Hence, it is applicable regardless of the prevailing modeling paradigm (imperative or declarative) and the control-flow complexity as it focuses on resources and activities, and not on the order of activities. Only when the arrival time of a case at an activity is not recorded and needs to be imputed, a control-flow notion (such as knowledge on the prior activity for a case) is required.

When applying the proposed method, the required computations to obtain resource availability calendars depends on several factors, including the following ones. Firstly, it depends on the size of the event log. The larger the event log becomes, the more activity instances need to be taken into consideration when applying the method. Secondly, more computations are required when resource availability calendars need to be created for more resources, i.e. when more distinct resources are included in the event log. Thirdly, the number of merged active periods will determine the number of times at which pending cases need to be identified. The number of merged active periods will be influenced by the characteristics of the process (e.g. the typical duration of activities) and the maximum time gap that the user allows between activity instances for merging purposes. Finally, the number of elements in the date classification mechanism will determine the required computations for the sampling approaches. This is especially relevant when cluster-based sampling is used to create resource availability calendars as each element of the date classification mechanism requires the creation of a clustering solution.

5.2. Generalizability

The primary use case for the resource availability calendars in this paper is the development of a process simulation model. However, this does not limit the generalizability of the proposed method. Inductive insights in the availability of a resource for a single business process can also be used to, e.g., shape the constraints in staff scheduling problems or to establish resource assignment rules. At a system design level, insights in the actual availability of resources can be used to configure, for instance, the time pattern called schedule restricted elements [55]. Depending on the use case, it might be useful to consider the full set of daily availability records (stage 1) as a starting point or the resource availability calendars containing sampled daily availability records (stage 2).

Another perspective on the proposed method's generalizability is that it can also be applied to determine resource availability for several or all processes of an organization. In that case, the event log used as an input should relate to all business processes that the user wants to take into consideration.

The potential of the method proposed in this paper can also be linked to the workflow resource patterns specified by Russell et al. [16]. The daily availability records and resource availability calendars can provide inductive insights in the workflow resource patterns. Consider, for instance, pull patterns highlighting how resources commit to perform a work item [16]. Using the methods in this paper, companies can retrieve inductive insights in the specific time intervals during which resources will take

Table 7
Summary statistics (mean/standard deviation/minimum/maximum) of (the absolute value of) the percentage difference between simulation output measures with real calendar and with sampled availability calendars (over all 20 simulations).

Output measure	Simulation output with real sch.	(abs. value of) pct. difference between simulation output with real calendar and sampled calendar	
		Direct sampling sch. mean/st.dev./min./max.	Cluster-based sampling sch. mean/st.dev./min./max.
Mean cycle time	138.62 h	3.90/2.62/0.05/11.26	4.84/3.21/0.06/11.44
Number proc. inst.			
- r1	3433	1.88/1.08/0.38/4.05	2.06/1.64/0.15/5.10
- r2	1930	2.15/1.14/0.05/3.99	3.69/2.01/0.21/7.20
- r3	2890	2.56/1.55/0.00/6.19	4.00/1.21/2.15/6.26
- r4	2179	3.15/1.13/1.47/5.83	2.21/0.99/0.55/3.85
- r5	746	1.96/1.69/0.00/6.03	2.75/1.53/0.40/6.16
- r6	4088	1.45/0.92/0.12/3.28	1.32/1.13/0.07/4.16
- r7	1290	10.85/3.95/4.26/17.13	14.43/2.88/9.37/19.61
n finished ent.	2582	2.83/2.22/0.00/6.82	3.35/2.11/0.00/7.09

up work items in the process. When these findings are linked to process performance analyses, e.g., showing which resources cause process delays, the specification of push patterns can also be supported. Push patterns indicate how the system presents or allocates work items to resources [16]. Inductive insights can determine, e.g., whether the system needs to prioritize particular work item for a resource at particular moments.

5.3. Limitations

Despite the potential of the proposed method, some limitations also need to be recognized. Firstly, this paper focuses on the temporal perspective of resource availability. However, it needs to be recognized that this is not the only determining factor of resource availability. Contextual factors, such as an unexpected increase in workload or the illness of a colleague, will also influence a resource's availability for a process. Such contextual factors can be captured by extending the current outputs with inductive resource availability rules.

Secondly, the application of the proposed method places some requirements on the event log used as an input. It is required that the event log contains a start and completion time for each activity instance. This is often not the case in real-life event logs. Moreover, the method uses the arrival time of a case at an activity (i.e. the time at which the case joins the queue of that activity). When the arrival time is not recorded by the system, it should be imputed in the event log using a heuristic. A simple heuristic involves assuming that a case arrives at an activity immediately after the prior activity has been completed. However, this heuristic might not be realistic in process contexts in which cases need to wait before proceeding to the next process step, even when the required resources are available.

Thirdly, when retrieving daily availability records, a resource is deemed to be unavailable for the process when he/she could process work items, but refrains from doing so. This assumes that a resource will be active when work items are present and he/she is available for the process. This assumption is rather rigid. It might not hold in reality as resources might not be eager to process particular work items, or might think it is more appropriate to defer them to a colleague. To anticipate upon this observation, a lower bound for the unavailability period duration can be enforced when post-processing the daily availability records.

Finally, the experimental design used to evaluate the sampling methods to create a resource availability calendar does not enable a structured comparative analysis between them. However, it should be recognized that this was not the goal of the conducted experiment. The goal was to investigate the ability of a sampling method to generate accurate simulation outputs when used as an input for a process simulation model. To conduct a full comparative analysis, an experimental set-up is required in which both sampling approaches are applied to a wide range of synthetic logs containing a variety of resource availability patterns (e.g. stable availability patterns versus very irregular availability patterns).

6. Conclusion

This paper presented and empirically evaluated a method to automatically retrieve resource availability calendars for a single process from an event log. These resource availability calendars express timeframes during which a resource can be allocated to activities in that process. They are the first to take into account (i) the temporal dimension of availability, i.e. the time of day at which a resource is available, and (ii) intermediate availability interruptions (e.g. due to a break). In this way, the inductive method provides data-driven support to gather profound insights in resource availability for a particular process. These insights can be used, amongst others, to shape constraints in staff scheduling problems using empirical data, or to define directly implementable availability calendars for process simulation purposes.

Potential directions for future research follow immediately from the aforementioned limitations of the current study. Firstly, future work can link resource availability to case attributes or variables expressing the system state. This can enable the discovery of rules underlying a resource's availability patterns. Secondly, the method's first stage can be extended with a transfer time model to accommodate processes where cases have to wait before proceeding to the next process step, even when the resources for this step are immediately available. Thirdly, research efforts can be directed towards identifying behavioral patterns of resources such as the diversion of particular work items to colleagues. Such patterns can enrich resource availability insights. Finally, a comparative analysis between the proposed sampling methods can be conducted to identify circumstances under which a particular sampling method is preferable. These insights would enable the method to recommend the most appropriate sampling approach for a particular context based on the insights gathered in stage 1 (e.g. the variability in the retrieved daily availability records).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Pseudocode of the developed algorithms

Algorithm 1 Daily availability record retrieval

```
Input: eventLog: an event log (list of complex objects representing events),
    controlFlowNotion: knowledge on the prior activity that is executed for a case to
    support arrival time imputation, tolerances: time tolerances for merging active
    periods, singleQueue: knowledge on whether single queues prevail in the process
Output: d: daily availability records (list of complex objects representing time
    periods during which a resource is available or unavailable)
 1: $eventLog \leftarrow$ ADDARRIVALEVENTS($eventLog$, $controlFlowNotion$)
    ▷imputes arrival events using knowledge on the prior activity executed for a case
 2: $a \leftarrow$ CONVERTTOACTIVITYLOG($eventLog$)
    ▷creates activity instances by mapping corresponding events
 3: $a \leftarrow$ SORTACTIVITYLOG($a$)
    ▷sorts rows in activity log based on variables in following order: resource, start
    timestamp and complete timestamp
 4: $n \leftarrow$ NUMBEROFROWS($a$)                ▷number of rows in activity log
 5: $d \leftarrow$ new List()   ▷creates list that will contain daily availability records as list
    elements
 6: for $i = 2$ to $n$ do                              ▷merge active periods
 7:     $tol \leftarrow$ GETTOLERANCE($tolerances$, $a[i].activity$, $a[i].resource$)
    ▷determines time tolerance used when merging active periods
 8:     if $a[i].resource == a[i-1].resource$ and
 9:         $a[i].start \leqslant a[i-1].complete + tol$ then
10:         $d \leftarrow$ APPEND($d$, $a[i].resource$, $a[i-1].start$, $a[i].complete$, $available$})
    ▷adds merged active period as available period to daily availability record list
11:     end if
12: end for
13: $d \leftarrow$ MERGEPERIODS($d$)
    ▷merges consecutive availability periods where the end of one period equals the
    start of the next period
14: $c \leftarrow$ NUMBEROFROWS($d$)     ▷number of availability periods in daily availability
    record list
15: for $j = 1$ to $(c-1)$ do                          ▷detect pending cases
16:     $p \leftarrow$ DETECTBOUNDARYPENDINGCASES($a$, $d[j].resource$, $d[j].end$, $singleQueue$)
    ▷returns activity instances that qualify as boundary pending cases at time $d[j].end$
17:     if $p$ is not empty then
18:         $d \leftarrow$ APPEND($d$, $d[j].resource$, $d[j].end$, $d[j+1].start$, $unavailable$})
    ▷adds unavailability period to daily availability record list when boundary pending
    cases are detected
19:     else                                     ▷detect intermediate pending cases
20:         $r \leftarrow$ DETECTINTERMEDIATEPENDINGCASES($a$, $d[j].resource$, $d[j].end$,
    $d[j+1].start$, $singleQueue$)
    ▷returns activity instances that are qualify as intermediate pending cases in time
    period between $d[j].end$ and $d[j+1].start$
21:         if $r$ is not empty then
22:             $m \leftarrow$ NUMBEROFROWS($r$)                ▷number of instances in r
23:             for $k = 1$ to $m$ do
24:                 if $r[k].resource == d[j].resource$ then
25:                     $d \leftarrow$ APPEND($d$, {$d[j].resource$, $r[k].arrival$, $d[j+1].start$,
    $unavailable$})
    ▷adds unavailability period to daily availability record list when intermediate
    pending cases is detected
26:                 else                   ▷will only be used when a single queue prevails
27:                     $endTime \leftarrow$ MINIMUM($r[k].start$, $d[j+1].start$)
    ▷determines end time of unavailability period, which is the earliest of the start of
    the intermediate pending case and the start of the next available period
28:                     $d \leftarrow$ APPEND($d$, {$d[j].resource$, $r[k].arrival$, $endTime$, $unavailable$})
    ▷adds unavailability period to daily availability record list for intermediate pending
    case
29:                 end if
30:             end for
31:         end if
32:     end if
33: end for
34: $d \leftarrow$ MERGEPERIODS($d$)
    ▷merges overlapping unavailability periods or consecutive unavailability periods
    where the end of one period equals the start of the next period correspond
35: $d \leftarrow$ SPLITPERIODS($d$)           ▷splits time periods crossing the day boundary
36: return $d$
```

Algorithm 2 Resource availability calendar creation — Direct sampling

```
Input: $d'$: daily availability records after post-processing (list of complex objects
    representing time periods in which a resource is available or unavailable), $r$:
    resource for which a resource availability calendar should be created, $c$: date
    classification mechanism, calStart: start date of the resource availability calendar
    that needs to be created, calEnd: end date of the resource availability calendar
    that needs to be created
Output: availCalendar: resource availability calendar (list of complex objects
    representing selected daily availability records)
 1: $dRes \leftarrow$ SELECTDAILYAVAILRECORDS($d'$, $r$)
    ▷returns daily availability records related to resource r
 2: $availCalendar \leftarrow$ new List()      ▷creates list that will contain sampled daily
    availability records as list elements
 3: $date \leftarrow calStart$                                   ▷initialize iterator
 4: while $date \leq calEnd$ do
 5:     $dRes' \leftarrow$ SELECTPOTENTIALDARS($d'$, $date$, $c$)
    ▷returns potential daily availability records taking into account the date
    classification mechanism
 6:     $selected \leftarrow$ RANDOMLYSELECTDAR($dRes'$)
    ▷returns one of the potential daily availability records (randomly selected)
 7:     $availCalendar \leftarrow$ APPEND($availCalendar$, $selected$)
 8:     $date \leftarrow nextDate$                                  ▷return next date
 9: end while
10: return $availCalendar$
```

Algorithm 3 Resource availability calendar creation — Cluster-based sampling

```
Input: $d'$: daily availability records after post-processing (list of complex objects
    representing time periods in which a resource is available or unavailable), $r$:
    resource for which a resource availability calendar should be created, $c$: date
    classification mechanism, calStart: start date of the resource availability calendar
    that needs to be created, calEnd: end date of the resource availability calendar
    that needs to be created
Output: availCalendar: resource availability calendar (list of complex objects
    representing selected daily availability records)
 1: $dRes \leftarrow$ SELECTDAILYAVAILRECORDS($d'$, $r$)
    ▷returns daily availability records related to resource r
 2: $n \leftarrow$ NUMBEROFELEMENTS($c$)      ▷return number of elements in date classification
    mechanism
 3: $clusterList \leftarrow$ new List()  ▷creates list that will contain a cluster solution for each
    element of the date classification mechanism
 4: for $i = 1$ to $n$ do                         ▷determine clustering solutions
 5:     $dResSubset \leftarrow$ SELECTDAILYAVAILRECORDS($dRes$, $c[i]$)
    ▷returns daily availability records related to a specific entry of the date classification
    mechanism
 6:     $clusterSol \leftarrow$ CREATECLUSTERS($dResSubset$)
    ▷returns cluster solution in the form of cluster medoids and cluster sizes
 7:     $clusterList \leftarrow$ APPEND($clusterList$, $clusterSol$)
 8: end for
 9: $availCalendar \leftarrow$ new List()      ▷creates list that will contain sampled daily
    availability records as list elements
10: $date \leftarrow calStart$                                   ▷initialize iterator
11: while $date \leq calEnd$ do
12:     $relevantClusterSol \leftarrow$ RETRIEVECLUSTERSOL($clusterList$, $date$)
    ▷returns relevant cluster solution
13:     $selected \leftarrow$ WEIGHTEDSELECTION($relevantClusterSol$)
    ▷returns one of the medoids from the cluster solution (taking into account cluster
    sizes)
14:     $availCalendar \leftarrow$ APPEND($availCalendar$, $selected$)
15:     $date \leftarrow nextDate$                                  ▷return next date
16: end while
17: return $availCalendar$
```

References

[1] J. Nakatumba, Resource-aware Business Process Management: Analysis and Support (Ph.D. thesis), Eindhoven University of Technology, 2013.

[2] H.A. Reijers, W.M.P. van der Aalst, The effectiveness of workflow management systems: predictions and lessons learned, Int. J. Inf. Manage. 25 (5) (2005) 458–472.

[3] W.M.P. van der Aalst, Business process simulation survival guide, in: J. vom Brocke, M. Rosemann (Eds.), Handbook on Business Process Management 1, Springer, Heidelberg, 2015, pp. 337–370.

[4] W.M.P. van der Aalst, J. Nakatumba, A. Rozinat, N. Russell, Business process simulation, in: J. vom Brocke, M. Rosemann (Eds.), Handbook on Business Process Management, Springer, Heidelberg, 2010, pp. 313–338.

[5] A.T. Ernst, H. Jiang, M. Krishnamoorthy, D. Sier, Staff scheduling and rostering: a review of applications, methods and models, European J. Oper. Res. 153 (1) (2004) 3–27.

[6] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, L. De Boeck, Personnel scheduling: a literature review, European J. Oper. Res. 226 (3) (2013) 367–385.

[7] P. De Bruecker, J. Van den Bergh, J. Beliën, E. Demeulemeester, Workforce planning incorporating skills: state of the art, European J. Oper. Res. 243 (1) (2015) 1–16.

[8] M. Defraeye, I. Van Nieuwenhuyse, Staffing and scheduling under nonstationary demand for service: a literature review, Omega 58 (2016) 4–25.

[9] M. Dumas, M. La Rosa, J. Mendling, H.A. Reijers, Fundamentals of Business Process Management, Springer, Heidelberg, 2013.

[10] D.M. McBride, The Process of Research in Psychology, Sage, Thousand Oaks, 2016.

[11] W.M.P. van der Aalst, Process Mining: Data Science in Action, Springer, Heidelberg, 2016.

[12] N. Martin, F. Bax, B. Depaire, A. Caris, Retrieving resource availability insights from event logs, in: Proceedings of the 2016 IEEE International Conference on Enterprise Distributed Object Computing, 2016, pp. 69–78.

[13] C. Cabanillas, M. Resinas, A. del Río-Ortega, A. Ruiz-Cortés, Specification and automated design-time analysis of the business process human resource perspective, Inf. Syst. 52 (2015) 55–82.

[14] Z. Huang, X. Lu, H. Duan, Resource behavior measure and application in business process management, Expert Syst. Appl. 39 (7) (2012) 6458–6468.

[15] W. Zhao, L. Yang, H. Liu, R. Wu, The optimization of resource allocation based on process mining, Lecture Notes in Comput. Sci. 9227 (2015) 341–353.

[16] N. Russell, W.M.P. van der Aalst, A.H.M. Ter Hofstede, D. Edmond, Workflow resource patterns: Identification, representation and tool support, Lecture Notes in Comput. Sci. 3520 (2005) 216–232.

[17] C. Cabanillas, Process-and resource-aware information systems, in: Proceedings of the 2016 IEEE International Conference on Enterprise Distributed Object Computing, 2016, pp. 1–10.

[18] A. Kumar, W.M.P. van der Aalst, H.M.W. Verbeek, Dynamic work distribution in workflow management systems: how to balance quality and performance, J. Manage. Inf. Syst. 18 (3) (2002) 157–193.

[19] A. Kumar, R. Dijkman, M. Song, Optimal resource assignment in workflows for maximizing cooperation, Lecture Notes in Comput. Sci. 8094 (2013) 235–250.

[20] J. Xu, C. Liu, X. Zhao, Resource allocation vs. business process improvement: How they impact on each other, Lecture Notes in Comput. Sci. 5240 (2008) 228–243.

[21] L.T. Ly, S. Rinderle-Ma, P. Dadam, M. Reichert, Mining staff assignment rules from event-based data, Lecture Notes in Comput. Sci. 3812 (2006) 177–190.

[22] Y. Liu, J. Wang, Y. Yang, J. Sun, A semi-automatic approach for workflow staff assignment, Comput. Ind. 59 (2008) 463–467.

[23] Z. Huang, X. Lu, H. Duan, Mining association rules to support resource allocation in business process management, Expert Syst. Appl. 38 (8) (2011) 9483–9490.

[24] S. Schönig, C. Cabanillas, S. Jablonski, J. Mendling, Mining the organisational perspective in agile business processes, Lect. Notes Bus. Inf. Process. 214 (2015) 37–52.

[25] Y. Liu, J. Wang, Y. Yang, J. Sun, A semi-automatic approach for workflow staff assignment, Comput. Ind. 59 (5) (2008) 463–476.

[26] G. Havur, C. Cabanillas, J. Mendling, A. Polleres, Resource allocation with dependencies in business process management systems, Lect. Notes Bus. Inf. Process. 260 (2016) 3–19.

[27] H. Yang, C. Wang, Y. Liu, J. Wang, An optimal approach for workflow staff assignment based on hidden Markov models, Lecture Notes in Comput. Sci. 5333 (2008) 24–26.

[28] H.A. Reijers, M. Jansen-Vullers, M. zur Muehlen, W. Appl, Workflow management systems + swarm intelligence = dynamic task assignment for emergency management applications, Lecture Notes in Comput. Sci. 4714 (2007) 125–140.

[29] P. Senkul, I.H. Toroslu, An architecture for workflow scheduling under resource allocation constraints, Inf. Syst. 30 (5) (2005) 399–422.

[30] R.S. Mans, N.C. Russell, W.M.P. van der Aalst, A.J. Moleman, P.J.M. Bakker, Schedule-aware workflow management systems, Lecture Notes in Comput. Sci. 6550 (2010) 121–143.

[31] I. Barba, B. Weber, C. Del Valle, A. Jiménez-Ramírez, User recommendations for the optimized execution of business processes, Data Knowl. Eng. 86 (2013) 61–84.

[32] N. Melão, M. Pidd, Use of business process simulation: a survey of practitioners, J. Oper. Res. Soc. 54 (1) (2003) 2–10.

[33] T. Baines, S. Mason, P.-O. Siebers, J. Ladbrook, Humans: the missing link in manufacturing simulation? Simul. Model. Pract. Theory 12 (7) (2004) 515–526.

[34] A. Rozinat, R.S. Mans, M. Song, W.M.P. van der Aalst, Discovering simulation models, Inf. Syst. 34 (3) (2009) 305–327.

[35] N. Martin, B. Depaire, A. Caris, The use of process mining in business process simulation model construction: structuring the field, Bus. Inf. Syst. Eng. 58 (1) (2016) 73–87.

[36] J.P. Cavada, C.E. Cortés, P.A. Rey, A simulation approach to modelling baggage handling systems at an international airport, Simul. Model. Pract. Theory 75 (2017) 146–164.

[37] T. Ziarnetzky, L. Mönch, A. Biele, Simulation of low-volume mixed model assembly lines: modeling aspects and case study, in: Proceedings of the 2014 Winter Simulation Conference, 2014, pp. 2101–2112.

[38] Z.S.-Y. Wong, A.C.-H. Lit, S.-Y. Leung, K.-L. Tsui, K.-S. Chin, A discrete-event simulation study for emergency room capacity management in a Hong Kong hospital, in: Proceedings of the 2016 Winter Simulation Conference, 2016, pp. 1970–1981.

[39] M. Leyer, J. Moormann, Comparing concepts for shop floor control of information-processing services in a job shop setting: a case from the financial services sector, Int. J. Prod. Res. 53 (4) (2015) 1168–1179.

[40] A. Wombacher, M. Iacob, M. Haitsma, Towards a performance estimate in semi-structured processes, in: Proceedings of the 2011 IEEE International Conference on Service-Oriented Computing and Applications, 2011, pp. 1–5.

[41] H.A. Reijers, J. Mendling, J. Recker, Business process quality management, in: J. vom Brocke, M. Rosemann (Eds.), Handbook on Business Process Management, Springer, Heidelberg, 2010, pp. 167–185.

[42] N. Damij, T. Damij, Process Management: A Multi-disciplinary Guide to Theory, Modeling, and Methodology, Springer, Heidelberg, 2014.

[43] OMG, Business Process Model and Notation (BPMN) Version 2.0, Tech. rep., 2011.

[44] T. Baier, J. Mendling, M. Weske, Bridging abstraction layers in process mining, Inf. Syst. 46 (2014) 123–139.

[45] A. Senderovich, M. Weidlich, A. Gal, A. Mandelbaum, Queue mining–predicting delays in service processes, Lecture Notes in Comput. Sci. 8484 (2014) 42–57.

[46] J. De Weerdt, M. De Backer, J. Vanthienen, B. Baesens, A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs, Inf. Syst. 37 (7) (2012) 654–676.

[47] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F.M. Maggi, A. Marrella, M. Mecella, A. Soo, Automated discovery of process models from event logs: review and benchmark, IEEE Trans. Knowl. Data Eng. 31 (4) (2018) 686–705.

[48] J.F. Hair, W.C. Black, B. Babin, R. Anderson, Multivariate Data Analysis, Pearson, Upper Saddle River, 2010.

[49] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Elsevier, Burlington, 2011.

[50] A. Bhat, K-medoids clustering using partitioning around medoids for performing face recognition, Int. J. Soft Comput. Math. Control 3 (3) (2014) 1–12.

[51] A.J. Izenman, Modern Multivariate Statistical Techniques. Regression, Classification and Manifold Learning, Springer, New York, 2008.

[52] A. Mooi, M. Sarstedt, A Concise Guide to Market Research, Springer, Heidelberg, 2014.

[53] B. Everitt, S. Landau, M. Leese, D. Stahl, Cluster analysis, Chichester, New York, 2011.

[54] P. Brereton, B.A. Kitchenham, D. Budgen, Z. Li, Using a protocol template for case study planning., in: EASE, Vol. 8, 2008, pp. 41-48.

[55] A. Lanz, M. Reichert, B. Weber, Process time patterns: a formal foundation, Inf. Syst. 57 (2016) 38–68.