Discovery and Simulation of Business Processes with Probabilistic Resource Availability Calendars

Orlenys López-Pintado and Marlon Dumas

August 25, 2023

Discovery and Simulation of Business Processes with Probabilistic Resource Availability Calendars

1st Orlenys López-Pintado
University of Tartu
Tartu, Estonia
orlenyslp@ut.ee

2nd Marlon Dumas
University of Tartu
Tartu, Estonia
marlon.dumas@ut.ee

Abstract—**In the field of business process simulation, the availability of resources is captured by assigning a calendar to each resource, e.g., Monday-Friday 9:00-18:00. Resources are assumed to be always available to perform activities during their calendar. This assumption often does not hold due to interruptions, breaks, or because resources time-share across multiple processes. A simulation model that captures availability via crisp time slots (a resource is either on or off during a slot) does not capture these behaviors, leading to inaccuracies in the simulation output. This paper presents a simulation approach wherein resource availability is modeled probabilistically. In this approach, each availability time slot is associated with a probability, allowing us to capture, for example, that a resource is available on Fridays between 14:00-15:00 with 90% probability and between 17:00-18:00 with 50% probability. The paper proposes an algorithm to discover probabilistic availability calendars from event logs. An empirical evaluation shows that simulation models with probabilistic calendars discovered from event logs, replicate the temporal distribution of activity instances and cycle times of a process more closely than simulation models with crisp calendars.**

Index Terms—**Business process simulation, process mining**

I. Introduction

Business Process (BP) simulation is a technique to predict how changes to a process will impact on its performance. It enables analysts to answer "what-if" questions such as "What would be the impact of switching some resources from full-time to part-time on the cycle time of the process?". The starting point of a BP simulation is a process model, represented for example in the Business Process Model and Notation (BPMN) [1], enhanced with simulation parameters such as the processing times of each activity, the inter-arrival time between cases, the available resources and their availability, etc. [2]. Each execution of a BP simulation model (a.k.a. a simulation run) produces an event log recording the simulated execution of a number of cases of the process, alongside aggregate performance statistics.

Resource availability plays a pivotal role in a BP simulation, as it determines the waiting times of the simulated activity instances. In mainstream BP simulation approaches, such as the BPSim standard specification [3], resource availability is captured by assigning a calendar to each resource or group of resources, e.g., from Monday to Friday, 9:00-18:00. This approach assumes that resources are available during every time slot in their calendar. In other words, these approaches interpret an availability calendar as a crisp set of time-slots: a resource is either available or not available during a given time-slot. In practice, this assumption often does not hold since the availability of resources is affected by interruptions, breaks, meetings, or time-sharing across multiple processes.

To tackle this limitation, this paper proposes: (1) a business process simulation approach wherein resource availability is captured via probabilistic calendars instead of crisp calendars; and (2) a method to discover such probabilistic availability calendars from event logs. In the proposed approach, an availability calendar associates a probability to each time slot in a calendar, e.g., a resource may be available with 90% probability on Fridays from 9:00-10:00, and this probability decreases linearly every hour down to 30% from 17:00-18:00 and then 5% from Friday 18:00 to Monday 8:00. At each time instant, the resource may be available or not, according to the probability of the corresponding time slot. The paper reports on an empirical evaluation aimed at testing the hypothesis that simulation models with probabilistic calendars discovered from event logs more closely replicate the temporal performance of a process (as recorded in the event log) than simulation models with crisp calendars.

The rest of the paper is structured as follows. Section II discusses related work. Section III describe and formalizes the probabilistic simulation approach. Section IV proposes the corresponding method to discover the simulation models. Section V empirically compares crisp vs. probabilistic simulation models, and Section VI concludes and sketches future work.

II. Related Work

Van der Aalst et al. [2], [4] discuss limitations of existing BP simulation approaches, including insufficient use of execution data in constructing simulation models and incorrect modeling of resources, particularly for resources that are not fully dedicated to one single process, but that, instead, time-share across multiple processes. In this paper, we address these limitations by proposing a BP simulation approach that uses probabilistic calendars to model resources that are intermittently available during certain time-slots, and an approach to discover such probabilistic calendars from execution data.

Freitas & Pereira highlight another common limitation of BP simulation modeling approaches, namely the assumption

Work funded by European Research Council (PIX project).

that all resources in a group (a.k.a., a resource pool) have the same availability calendar [5]. Some tools like IBM Websphere Modeler[1] support named resources, each having its own availability calendar. In this paper, we adopt this latter approach (calendars may be attached to each resource individually), but we extend it to support probabilistic availability calendars.

Discrete-event simulation tools like Arena [6] can model probabilistic availability through resource failure models, which allows for resource availability scheduling and incorporating potential failures and downtimes relying on activity instances. This paper adopts a more direct approach in which resource availability is always probabilistic, independent of any activity instance or failures. We also address the problem of discovering probabilistic calendars from event logs.

Several studies have proposed methods to discover BP simulation models from event logs. The first generation of studies in this direction [7]–[9] assumed that resources are continuously available (24/7). More recent studies [10], [11] incorporate algorithms for discovering resource calendars from event logs. In our earlier work, we extended these approaches to discover availability calendars for individual resources (instead of resource pools) [12] and implemented it in the Prosimos open-source BP simulator [13]. However, the above approaches discover crisp availability calendars, i.e., a resource is either available or not during each time-slot in the calendar.

The problem of representing probabilistic (or fuzzy) calendars is examined in [14]. The authors define fuzzy calendars as functions that assign a probability to each interval in a set of intervals (e.g., every Monday at 9-10 am during the year 2023). The authors propose a fuzzy algebra for defining more complex fuzzy calendars. Our approach takes the idea of a probabilistic calendar as a mapping from (periodic) sets of intervals to probabilities, and incorporates it into a BP simulation approach.

III. Simulation with Probabilistic Calendars

We take as a starting point a resource model wherein each resource has a *resource profile*. The profile of a resource determines the activities that the resource may perform, the performance of this resource (i.e., how much time the resource takes to perform different activities), the cost per time unit of the resource, and the availability calendar of a resource. This model treats resources in a differentiated manner, insofar as each resource has its own profile, but it does not prevent multiple resources to have identical profiles. An activity may be performed by more than one resource, and multiple activities may share common resources. The time that a resource takes to perform an activity is captured as a number drawn from a distribution. Availability calendars consist of the time intervals during which a resource is available, under the assumption that a resource is either available continuously or not available during each of these intervals (i.e., crisp calendar). These considerations are formalized below.

Definition 1 (BP simulation model with differentiated resources - from [12]): A BP simulation model with differentiated resources DSM is a tuple $< E, A, G, F,$ $\textsc{RProf}, BP, AT, AC >$, where E, A, G are the sets of events, activities, and gateways of a BPMN model, F is the set of directed flow arcs of a BPMN model, and the remaining elements capture simulation parameters as follows:

1) $\textsc{RProf} = \{r_1, ..., r_n\}$ is a set of resource profiles, where n is the number of resources in the process, and each resource $r \in RProf$ is described by:
   - $\textsc{Alloc}(r) \subseteq A$ is the set of activities that r can execute
   - $\textsc{Perf}(r, \alpha) = \textsc{Alloc}(r) \to PDF$ is a function that maps each activity $\alpha \in \textsc{Alloc}(r)$ to a probability density function $\mathcal{P} \in PDF$ with image over positive real numbers, representing the distribution of the processing times of activity α when assigned to r
   - $\textsc{Avail}(r)$ is the calendar (set of intervals) in which resource r is available to perform activities in $\textsc{Alloc}(r)$
   - $\textsc{Cost}(r)$ is the unit cost (e.g. per hour) of resource r
2) BP: $F \to [0, 1]$ is a function that maps each flow $f \in F$ s.t., the source of f is an element of G to a probability (a.k.a., the branching probability).
3) AT $\in \mathcal{P}(\mathbb{R}+)$ is a probability density function modeling the inter-arrival times between consecutive case creations.
4) AC is a calendar (set of intervals) such that cases can only be created during an interval in AC.

A Business Process (BP) simulation model, as in Definition 1 produces an event log when executed in a simulation engine. An event log consists of events - instances of activities of a process. Each event contains an activity label, the resource that performed the activity, and datetimes[2] marking when the activity was enabled, started, and ended. Traces are non-empty sequences of events. An event log is a set of traces, each representing a process instance (case). We write 'simulated log' to refer to a log produced by a simulation model, and 'real log' to refer to a log extracted from an information system. Several performance metrics can be derived from these logs to assess process efficiency, for example, *waiting time* – the time-span from enabling time to the start of the event; *processing time* – the time-span between beginning and end of the event; *cycle time* – the time-span between the enabling time and end of an event; and *resource utilization* – the ratio between the time a resource is busy executing activity instances, divided by its total availability time.

This paper focuses on discovering and representing the functions $\textsc{Avail}$ and $\textsc{Perf}$ in Definition 1. Accordingly, Definitions 2, 3, 4 formalizes the concepts of time granularity, probabilistic granularity, and probabilistic resource calendar proposed in this paper to model $\textsc{Avail}$. The $\textsc{Perf}$ function is modeled by $\textsc{adjustProcessingTime}$ in Definition 4.

Time Granularity (Definition 2) refers to segmenting time into defined, discrete intervals, often called 'granules'. Each

[1]https://www.ibm.com/support/pages/download-websphere-business-modeler-advanced-v70

[2]We write *timestamp* to refer to a time relative to a start of a day, e.g., 13:00:00 and *date-time* to refer to a time including a date and time of day.

granule has the same duration (expressed in some time unit, e.g., seconds, minutes, hours), and these intervals do not overlap or intersect. So, granules provide the minimum interval for positioning each event datetime executed in a process. Then, Definition 3 assigns probability values to time granules within recurring slots. For example, assuming the days of the week as recurrent slots, a probabilistic granularity represents cases like a resource working every Monday (recurrent slot) from 9:00-10:00 (granule of size 1 hour) with a probability of 0.5. Thus, it models the availability not deterministically but instead influenced by various probabilistic factors. Finally, Definition 4 combines the ideas of time granularity, probabilistic granularity, and methods to model resource availability by probabilistic calendars.

Definition 2 (Time Granularity): A time granularity $\Delta_{\tau_0,d,n}$ is a sequence of consecutive timestamps $\tau_0, \tau_1, ..., \tau_{n+1}$, defined by the tuple $< \tau_0, d, n >$. So, two consecutive timestamps in Δ define a time granule, $\delta_i = [\tau_i, \tau_{i+1})$, which is a time interval of duration d, with n being the total number of time granules in Δ. Besides, the following rules must hold: $\forall \tau_i \in \Delta, \tau_i = \tau_0 + (i * d)$, $\tau_i - \tau_{i-1} = d$.

Definition 3 (Probabilistic Granularity): Let Ω be a set of recurring slots and a time granularity Δ. A probabilistic granularity is a function $P : \Omega \times \Delta \rightarrow [0, 1]$ mapping each pair $(\omega \in \Omega, \delta \in \Delta)$, named p-granule, to a real number in [0..1] representing a probability. Recurring slots, $\omega \in \Omega$ are defined by a unique identifier and a period strictly determined by the first and last timestamps $[\tau_0, \tau_{n+1}]$ in granularity Δ.

Definition 4 (Probabilistic Resource Calendar): A probabilistic resource calendar consists of the following functions:

- ABSOLUTEPROBABILTY, P_{ABS}, is a probabilistic granularity that quantifies the probability of a resource being available in a p-granule, given that a task that can be allocated to them is enabled.
- RELATIVEPROBABILTY, P_{REL}, is a probabilistic granularity that quantifies the probability of a resource to be available in a p-granule $< \omega, \delta >$ relative to how often other resources are available in $< \omega, \delta >$.
- Γ is a function that retrieves the corresponding recurring slot and time granule from a given datetime.
- ISAVAILABLE is a function that, given a p-granule $< \omega, \delta >$, applies Bernoulli distributions to decide whether the resource is available (or not):
  (`True` with $P_{ABS}(\omega, \delta)$, `False` with 1 - $P_{ABS}(\omega, \delta)$) **or**
  (`True` with $P_{REL}(\omega, \delta)$, `False` with 1 - $P_{REL}(\omega, \delta)$)
- NEXTAVAILABLETIME is a function that, given a datetime σ, retrieves the nearest datetime σ' in which the resource will be available as follows: $\min \sigma' : \sigma' \geq \sigma \land$ ISAVAILABLE$(\Gamma(\sigma')) =$ `True`.
- ADJUSTPROCESSINGTIME is a function that receives a datetime σ and a floating number pt representing an ideal processing time, i.e., assuming the resource is fully dedicated and available during that pt period. Then, it returns a datetime σ' after adjusting pt by adding the time the resource is unavailable according to their calendar, i.e., $\sigma' = \sigma + pt + \sum_{\delta}^{isAvailable(\delta)=False} |\delta|$, $\delta \in \{\Gamma(\sigma_{i+1}) : \sigma_{i+1} = \sigma_i + d\}_{\sigma}^{\sigma'}$, being d the time interval duration defined by the corresponding granularity.

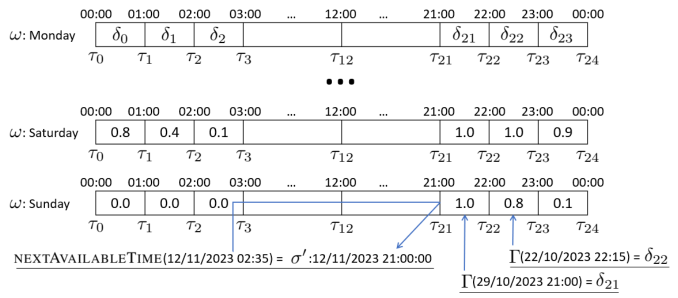
00:00 01:00 02:00 03:00 ... 12:00 ... 21:00 22:00 23:00 00:00
ω: Monday
δ_0 δ_1 δ_2 δ_{21} δ_{22} δ_{23}
τ_0 τ_1 τ_2 τ_3 τ_{12} τ_{21} τ_{22} τ_{23} τ_{24}
ω: Saturday
0.8 0.4 0.1 1.0 1.0 0.9
ω: Sunday
0.0 0.0 0.0 1.0 0.8 0.1
NEXTAVAILABLETIME(12/11/2023 02:35) = σ':12/11/2023 21:00:00
Γ(22/10/2023 22:15) = δ_{22}
Γ(29/10/2023 21:00) = δ_{21}

Fig. 1: Probabilistic Weekly Calendar with granularity $\Delta = (\tau_0 = 00:00:00, d = 1\ hour, n = 24\ hours)$.

Figure 1 sketches the time granules for a probabilistic weekly calendar. In this calendar, the recurrent slots are the weekdays, i.e., they repeat every seven days, each split into n = 24 granules ($\delta_0, ..., \delta_{23}$), each of size d = 1 hour, and starting at midnight $\tau_0 = 00:00:00$. Granules on Saturday and Sunday have an associated probability value (p-granules), for example, pointing out that the resource is always available on Saturdays from 21:00 to 23:00 and never on Sundays from 00:00 to 03:00. The bottom of Figure 1 illustrates how the function Γ retrieves the corresponding p-granules given two datetimes, i.e., both correspond to a Sunday, but to the granules δ_{21} and δ_{22} in which the resource is available with a probability of 1.0 and 0.8 respectively. Finally, the function NEXTAVAILABLETIME probabilistically determines that if requested on 12/11/2023 at 02:35, the resource will be available the same day but at 21:00 at the earliest. Note that although not illustrated in the figure, function NEXTAVAILABLETIME may retrieve a different (future) date, i.e., one that does not correspond to the input datetime.

Characterizing a resource calendar by combining absolute (P_{ABS}) and relative (P_{REL}) probabilities in Definition 4 deserves further explanation. On the one hand, P_{ABS} measures availability by counting the frequency ratio of a resource from all the occurrences of related events in a given p-granule. However, a uniform task allocation to resources may negatively impact this value. For example, assume a task always scheduled on Mondays from 9:00-10:00 and a pool of 10 resource candidates. In this case, if a different resource is appointed alternately every Monday, each will exhibit a low probability of 0.1 after the first rotation, even when they were always available. In cases like that, a relative value comparing the resources with the busiest one in the granule leads to a more accurate estimation. On the other hand, P_{REL} evaluates resource frequency relative ratio to the most occupied resource in a granule, potentially disadvantaging resources with lighter workloads. For example, a frequently allocated resource within a given p-granule could substantially lower the probability of other resources that were consistently available but not needed due to the lack of enabled tasks. Consequently, our method combines absolute and relative probabilities to capture a more comprehensive range of resource availability.

Algorithm 1 Probabilistic Resource Allocation

```
1: function AllocateResource(e: EnabledEvent, rQ: PriorityQueue, pt)
2:     r, $\sigma \leftarrow$ popMin(rQ)
3:     if enabledAt[e] $> \sigma$ then
4:         $\sigma \leftarrow$ nextAvailableTime($\sigma$)
5:     startedAt[e] $\leftarrow \sigma$
6:     completedAt[e] $\leftarrow$ adjustProcessingTime($\sigma$, pt)
7:     Enqueue(rQ, r, nextAvailableTime( completedAt[e]))
```

Algorithm 1 illustrates the overall semantics for allocating a resource to an enabled event e when simulating a process modeled by Definitions 1- 4. All the resources are stored by the following datetime in which they will be available (or operational) in a priority queue rQ. Thus, line 2 procures the earliest active resource, r, alongside its corresponding available datetime, σ. Although r is operational at σ, the event enablement datetime may occur in a granule posterior to the one of σ. Therefore, lines 3-5 ensure the resource is not assigned (and starts) a task before its enablement. Then, line 6 computes the event completion datetime by adjusting the preliminary (under ideal conditions) processing time pt as r might be unavailable on some of the intervals spanned by pt. Finally, line 7 reinserts r to the queue rQ with a new datetime corresponding to the next available time after completing the current event. Note that function nextAvailableTime is stochastic, which implies it may return different granules if invoked several times from the same datetime. Therefore, since the resources in rQ are already marked as available, it is not advisable to invoke the functions nextAvailableTime or isAvailable for the same granule because they execute a coin flip again, potentially altering the status of the granule from available to unavailable and vice versa.

IV. Discovering Probabilistic Resource Calendars

Algorithms 2-4 describe our approach to discovering differentiated probabilistic calendars, which model resource availability from an event log. This approach considers the precise datetime when tasks are enabled and executed and the resources implicated in each task. To accomplish this, Algorithm 2 receives an event log L, a time duration d' in minutes, and an angle $\beta \in [0, 1]$ as inputs. The angle β will be instrumental in determining the probabilities within each granule. Initially, lines 2-6 map every resource within L to a granularity Δ, starting at δ_0 = 0 (equating to midnight) with the input duration d' and a total number of granules $n = 1440//d'$, i.e., spanning the entire day (1440 minutes). For example, if d' is 60 minutes, the probabilistic calendars of each resource will have 24 granules (i.e., $\Delta = \delta_0, ..., \delta_{23}$) starting at midnight for every day of the week. More specifically, the matrices λ and Λ tally, respectively, the frequency at which a resource r was detected to be operational (λ) and when it was required (Λ) within a specific time granule δ on a day of the week ω in the log L. The matrix M counts the frequency at which the most utilized resource was operational within each p-granule.

Subsequently, Algorithm 2 in lines 7-11 computes the enabling times for each event within each trace T in the log and applies the trapezoidal method as detailed in Algorithm 3

Algorithm 2 Discovery of Probabilistic Calendars

```
1: function DiscoverIntervals($L$: EventLog, d': Minutes, $\beta$: angle)
2:     for each day of week $\omega \in \{Monday, ..., Sunday\}$ do
3:         $M[\omega] \leftarrow \Delta(\tau_0 = 0, d = d', n = 1440//d')$
4:         for each resource $r \in L$ do
5:             $\lambda[r][\omega] \leftarrow \Delta(\tau_0 = 0, d = d', n = 1440//d')$
6:             $\Lambda[r][\omega] \leftarrow \Delta(\tau_0 = 0, d = d', n = 1440//d')$
7:     for each trace $T \in L$ do
8:         ComputeEnablingTimes($T$)
9:         for each event $e \in T$ do
10:            Trapezoidal(e, $\lambda$, $\Lambda$, $M$, $\beta$, False)
11:            Trapezoidal(e, $\lambda$, $\Lambda$, $M$, $\beta$, True)
12:    for each resource $r \in L$ do
13:        for each day of week $\omega \in \{Monday, ..., Sunday\}$ do
14:            for each time granule $\delta \in \Delta(\tau_0 = 0, d = d', n = 1440//d')$ do
15:                $P_{ABS} \leftarrow \lambda[r][\omega][\delta]$ / $\Lambda[r][\omega][\delta]$
16:                $P_{REL} \leftarrow \lambda[r][\omega][\delta]$ / $\Lambda[\omega][\delta]$
17:    return $P_{ABS}$, $P_{REL}$
```

Algorithm 3 Availability Calculation- Trapezoidal Method

```
1: function Trapezoidal($e$, $\lambda$, $\Lambda$, $M$, $\beta$, allocated)
2:     if allocated then
3:         $GR \leftarrow$ ExtractTimeGranules(startedAt[$e$], completedAt[$e$])
4:     else
5:         $GR \leftarrow$ ExtractTimeGranules(enabledAt[$e$], startedAt[$e$])
6:     if $|GR| = 1$ then
7:         UpdateGranules(1.0, $e$, $\delta_0 \in GR$, $\delta_0 \in GR$, $\lambda$, $\Lambda$, $M$, allocated)
8:     else
9:         $p \leftarrow 1.0$
10:        $f \leftarrow$ 1.0 / ($|GR|$ // 2) * $\beta$) if $\beta > 0$ else 1.0
11:        $s \leftarrow 0$, $e \leftarrow |GR| - 1$
12:        while $\delta_s < \delta_e$ do
13:            UpdateGranules($p$, $e$, $\delta_s \in GR$, $\delta_e \in GR$, $\lambda$, $\Lambda$, $M$, allocated)
14:            $p \leftarrow p - f$
15:            $s \leftarrow s + 1$, $e \leftarrow e - 1$
```

to every event e. Finally, in lines 12-16, for each resource in the event log and each day of the week, Algorithm 2 calculates the absolute and relative probabilities for each time granule, which are returned later in line 17. Specifically, the absolute probability P_{ABS} computes the ratio of the frequency at which a resource was operational to the total frequency at which it was required for an activity. Note that the required intervals also count the instances when the resource was operational. The relative probability P_{REL} measures the ratio between λ but divided by the most frequent resource in the corresponding granule. As such, the most frequently engaged resource in a given p-granule would possess a relative probability of 1.0.

Algorithm 3 describes a trapezoidal method to measure the resource availability within p-granules related to a given event, distributing the availability in a trapezoidal shape over the event duration. So, p-granules containing the event's start and end datetimes check the resource as operational. In contrast, the resource status is vague in the remaining granules between these points. The trapezoidal method tackles this uncertainty by assigning a weight of 1.0 to granules matching an event's start and end datetimes. Then, it reduces the weight of resource availability in proportion to the time distance of a p-granule from the empirically confirmed operational p-granules.

Algorithm 3 takes six parameters as input: an event e, the matrices λ, Λ and M, to be updated, the angle β shaping the trapezoid, and a boolean variable, *allocated*, which defines two types of intervals. The first type (False) corresponds to the interval between enabling and starting times of the event, a period in which the resource is required but not yet available.

Algorithm 4 Update Time Granule Availability Weights

```
1: function UpdateGranules($p$, $e$, $\delta_s$, $\delta_e$, $\lambda$, $\Lambda$, $M$, allocated)
2:     for $\delta \in \{\delta_s, \delta_e\}$ do
3:         $\omega \leftarrow$ DayOfWeek($\delta$, $e$)
4:         for each $rCand \in$ taskResources[task[$e$]] do
5:             if not isBusy[$rCand$][DateTime($\delta$)][$\delta$] then
6:                 $\Lambda[rCand][\omega][\delta] \leftarrow +1$
7:         if allocated then
8:             $\lambda$[resource[$e$]][$\omega$][$\delta$] $\leftarrow +p$
9:             $\Lambda$[resource[$e$]][$\omega$][$\delta$] $\leftarrow +1$
10:            $M[\omega][\delta] \leftarrow$ max($M[\omega][\delta]$, $\lambda$[resource[$e$]][$\omega$][$\delta$])
```

The second type (True) refers to the interval between starting and ending times of the event, a period in which the resource might be operational executing the related task.

Algorithm 3 operates as follows: lines 2-5 extract the sequence of granules according to the *allocated* parameter types. Then, lines 6-15 differentiate two cases to update the availability. If the duration of the event spans only one granule, then it is weighted twice with 1.0, as the resource started and ended the event within it (lines 6-7). When the event spans multiple granules, the initial weight is 1.0 at the boundaries and decreases towards the center (lines 9-15). The rate of this decrement relies on the angle β as described by the formula on line 10. A $\beta = 0.0$ assigns a weight of 1.0 to the boundary granules and 0.0 to the remaining ones. In contrast, a $\beta = 1.0$ decreases the weight of each granule in the sequence by a factor f given by $1.0/(|GR|//2) * \beta)$. For example, given five granules, $\beta = 0.0$ results in [1.0, 0.0, 0.0, 0.0, 1.0], and $\beta = 1.0$ in [1.0, 0.5, 0.0, 0.5, 1.0].

Algorithm 4, called in lines 7 and 13 of Algorithm 3, updates the granules determined by the trapezoidal method. This algorithm requires eight input parameters: the availability weight p, computed from β, two granules δ_s and δ_e that are at equal distance from the interval boundaries, the matrices λ, Λ, and M, and the boolean variable *allocated*.

For each of the two granules δ received, Algorithm 4 proceeds as follows: Line 3 identifies the day of the week on which δ occurs. Subsequently, the loop in lines 4-6 increments by one the frequency of pair $< \omega, \delta >$ in the matrices Λ for each non-operational resource capable of performing the task defined by the event e. This adjustment relies on the idea that all available resources potentially suited to the task were considered eligible but were ultimately not selected. This modification applies to granules within both interval types, from the event's enablement to start and from start to end. Then, lines 7-10 update p-granules within the intervals where a resource might be operational, i.e., engaged in the event execution from start to end. Thus, the matrix λ of the resource executing e is updated with the estimated availability factor p. The frequency of $< \omega, \delta >$ in the matrix Λ increments by 1, acknowledging that the resource was indeed required. Finally, the algorithm checks and updates whether a new, most frequent resource has emerged for that p-granule in matrix M.

When processing the event log, the allocated resources would ideally work on the corresponding tasks without interruption. However, according to their calendars, they can have some resting time in between. Accordingly, Algorithm 5 recalibrates the processing times observed in the event log

Algorithm 5 Fit Processing Times to Probabilistic Granules

```
1: function FitProcessingTimes($L$: EventLog, $P_{ABS}$, $P_{REL}$, $\kappa$)
2:     $adjTimes \leftarrow [resources \in L, tasks \in L]$
3:     $resourceTaskEvents \leftarrow$ GroupEventsByPairResourceTask($L$)
4:     for each $r, t \in resourceTaskEvents$ do
5:         for each event $e \in resourceTaskEvents[r][t]$ do
6:             $GR \leftarrow$ ExtractTimeGranules(startedAt[$e$], completedAt[$e$])
7:             $pTime \leftarrow 0$
8:             for each $\delta_i \in GR$ do
9:                 $\omega \leftarrow$ DayOfWeek($\delta$, $e$)
10:                if $i = 0$ or $i = n$ then
11:                    $d' \leftarrow +(\delta_i, e)_i * \max(P_{ABS}[r][\omega][\delta_i], P_{REL}[r][\omega][\delta_i])$
12:                else
13:                    $d' \leftarrow +d_{\delta_i} * \max(P_{ABS}[r][\omega][\delta_i], P_{REL}[r][\omega][\delta_i])$
14:            Append($adjTimes[r][t]$, $d'$)
15:    $pTimeDistr \leftarrow [resources \in L, tasks \in L]$
16:    for each $r, t \in resourceTaskEvents$ do
17:        if $|adjTimes[r][t]| \geq \kappa$ then
18:            $pTimeDistr[r][t] \leftarrow$ BestFitDistribution($adjTimes[r][t]$)
19:        else
20:            $pTimeDistr[r][t] \leftarrow \emptyset$
21:    for each $r, t \in resourceTaskEvents$ do
22:        if $pTimeDistr[r][t] = \emptyset$ then
23:            $rCandidate \leftarrow$ FindClosestMeanCandidate($t$)
24:            if $rCandidate \neq \emptyset$ then
25:                $pTimeDistr[r][t] \leftarrow pTimeDistr[rCandidate][t]$
26:            else
27:                $pTimeDistr[r][t] \leftarrow$ BestFitDistribution($\forall\ t \in L$)
28:    return $pTimeDistr$
```

to match granular time intervals identified by the trapezoidal method. The input of the algorithm consists of the event log L, the absolute and relative probability functions built by Algorithms 2-4, and a float number κ referring to the minimum frequency threshold for a resource-task pair occurrence needed to estimate their adjusted processing times.

This paper follows the differentiated performance model presented in [12]. Thus, the processing times are calculated for each pair resource task in L. Accordingly, lines 2-3 of Algorithm 5 initialize the $adjTimes$ matrix to tally the adjusted times and group the events in L by resource-task pairs. Lines 4-14 iterate over each event e for each pair resource r, task t. First, it extracts the granules δ_i from the start to the end of e (i.e., representing the processing time) and identifies each granule's corresponding weekday ω. Subsequently, the adjusted processing time is the sum of each granule duration (d_{δ_i}), each multiplied by their maximum probability between P_{ABS}, P_{REL}. The rationale is that granules are allocated selectively from their probabilities. Thus, multiplying each granule duration by the corresponding probability factor maintains or decreases the overall processing time, which, on average, may converge to the actual operational times. The first and last granules might only be partially covered. Thus, the notation $(\delta_i, e)_i$, with $i = 0, n$ represents the actual durations from the event's start to the end of the first granule and from the start of the last granule to the event's end.

After adjusting the processing times observed in L according to the probabilistic calendars, Algorithm 5 calculates a distribution function to model them. In this regard, for every resource-activity pair (lines 15-20), the algorithm ensures the number of processing times calculated fulfills the minimum level of significance settled by the parameter κ. If this condition is satisfied, the function BestFitDistribution constructs a histogram from them. It employs curve-fitting techniques to identify a probability distribution that offers the most precise approximation to the histogram, i.e., the one with the lowest residual sum. In cases where the pairs do not meet the κ requirement, the distribution function is built by aggregating (lines 21-27). Specifically, if other resources with an associated distribution executing the corresponding task exist, the algorithm assigns the distribution function of that resource with the closest mean to the non-adjusted processing times. Otherwise, the distribution aggregates the processing times observed for all the resources that performed the task.

V. Implementation and Evaluation

We implemented the proposed approach by extending the Simod tool for automated discovery of simulation models from event logs [9][3] and the Prosimos simulation engine [12], [13][4] to support probabilistic calendars. Using the extended versions of Simod and Prosimos, we conducted an empirical evaluation aimed at answering the following question: *Does the use of probabilistic calendars improve the accuracy of business process simulation models discovered from event logs, compared to crisp calendars?*

A. Datasets and Experimental Setup

The evaluation relies on eight synthetic and four real-life logs. To generate the synthetic logs, we took as starting point a simulation model of a loan application process.We generated eight event logs based on this process model. The first four, denoted by the prefix B in Table II and related to the model *Loan-B* in Table I, were generated by assuming resources with balanced workloads, i.e., allocating the tasks equally among resources playing the same role. The remaining logs, denoted by the prefix U in Table II and derived from *Loan-U* in Table I, were constructed assuming resources with unbalanced workloads. Task allocation to resources was unequal, with resources in each role exhibiting six different workloads, i.e., from a batch of 21 tasks, each resource received 6, 5, 4, 3, 2, and 1, respectively. The underlying reasoning is to model practical scenarios where resources may share a calendar yet have different workloads.

We constructed calendars for each balanced/unbalanced group as follows: (1) B-24, U-24 - all resources operate 24 hours, 7 days a week; (2) B-8, U-8 - resources work an 8-hour shift from Monday to Friday, with working hours from 8:00-11:00 and 13:00-17:00; (3) B-8/4, U-8/4 - half the resources work the 8-hour shift outlined in (2), with the remaining half working part-time, 4 hours from 13:00-17:00, from Monday to Friday; and (4) B-24/A, U-24/A - resources are divided equally among the 8-hour shift described in (2), the 4-hour shift as described in (3), and a schedule where resources work 24 consecutive hours followed by a 48-hour rest period.

The task-resource allocation was systematically rotated among resources within each role, maintaining the balance/unbalance ratios previously described. Similarly, the assigned calendars were rotated among the resources within each role. We then assigned each task-resource pair to a different distribution function to model the processing times. Finally, we used the resulting simulation models to generate synthetic logs using the Apromore simulator. We abstained from using Prosimos (and its associated queuing mechanism) for synthetic log generation to prevent potential bias, as we later use Prosimos to evaluate the models discovered by our approach. Each synthetic log contains 2000 cases with inter-arrival times following a Poisson distribution with a mean of 20 cases from 8:00 to 17:00, Monday to Friday.

The real-life logs in Table I have different complexities. The first one is a subset of the *BPIC-2012* log[5] – of a loan application process from a Dutch financial institution. We extracted the subset of this log consisting of activities with start and end timestamps. Similarly, we used the equivalent subset of the *BPIC-2017* log[6] - an updated version of the *BPI-2012* log. We extracted the subsets of the BPI-2012 and BPI-2017 logs by following the recommendations of the winning teams of the *BPIC-2017* challenge.[7] The third log *AC-CRE* is an anonymized log of an academic recognition process at a university. It contains a high number of resources with low participation in the process. The fourth log, *CALL*, is from a call center process including a high volume of cases of short duration, with an average of two activities per case.

For the real-life logs, we did not have a (BPMN) process model as a starting point (only the log). Given that our approach is independent of the algorithm used for process model discovery, we did not generate the models from these real-life logs automatically. Instead, we discovered the BPMN models from the event logs using the Apromore platform[8] and manually adjusted the discovered models until reaching a replay-based fitness [15] of 90%. The rationale for adjusting the models to have similar fitness levels is to reduce the impact of control-flow-related inaccuracies during the discovery of the simulation model, given that the accuracy of the branching probabilities computed by Simod depend on the ability to replay the traces in the log against the model.

TABLE I: Characteristics of the event logs used for evaluation.

	Loan-B/	**Loan-U/**	**BPIC12/**	**BPIC17/**	**AC-CRE/**	**CALL/**
Traces	2000	2000	8616	30276	954	521779
Events	23332	23288	59301	240854	6829	900374
Activities	17	17	6	8	18	19
Resources	54	54	58	148	561	3021

Table I gives descriptive statistics of the logs, including the number of traces, events, activities, and resources. The first two columns list the characteristics of the synthetic logs (the four balanced ones, Loan-B) and the four unbalanced ones (Loan-U). The remaining columns describe the real-life logs.

To avoid data leakage and overfitting, we split each log into two sets (training and testing) using a temporal split:

[3]https://github.com/AutomatedProcessImprovement/Simod
[4]https://github.com/AutomatedProcessImprovement/Prosimos
[5]https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
[6]https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
[7]https://www.win.tue.nl/bpi/2017/challenge.html
[8]https://apromore.com

the first 50% of traces in chronological order are put in the training dataset (to discover a simulation model) and other 50% are used for measuring the accuracy of the model. We evaluate accuracy of each simulation model by simulating it using PROSIMOS, and measuring the disparity between the simulated output and the testing subset of the log.

Chapela-Campa et al. [16] propose multiple metrics to evaluate the quality of data-derived simulation models, along the control flow, temporal, and congestion dimensions. Since our study focuses on resource availability, we only consider temporal and congestion metrics. In the congestion dimension, [16] proposes two metrics: one for arrival times and one for cycle times. We discard the inter-arrival metric (out of the scope). Instead, we take the arrival times from the testing log, so that errors in the estimation of arrival times do not affect the results. Thus, we only retain the Cycle Time Distribution (CTD) metric. On the temporal dimension, [16] proposes three metrics: Circadian Event Distribution (CED), Absolute Event Distribution (AED), and Relative Event Distribution (RED). Since our approach does not estimate inter-arrival times or seasonality, we omit the AED and CED metrics and include the RED metric, which captures the ability of the simulation model to replicate the occurrence of events (and their datetimes) from the beginning of a case to its end.

The CTD metric evaluates the simulation model's capacity to replicate the overall cycle time of a process. It derives empirical Probability Distribution Functions (PDF) from histograms of cycle times collected from two event logs, *L1* and *L2*, and calculates the CTD distance as the 1-Wasserstein Distance (1WD) [17] between these histograms. The Relative RED analyses the ability of the simulator to mimic the temporal distribution of events relative to the origin of the case. It adjusts all datetimes in *L1* and *L2* to originate from their respective case arrival times (i.e., the initial datetime in a case is set to 0, with the following datetimes adjusted by the inter-event times). The RED Distance computes the 1WD between the discretized event logs *L1* and *L2*.

Additionally to the RED and CTD, we measured the mismatch ratio (MMR) to determine the resource discrepancy between the real and simulated logs, i.e., $MMR = 1 - |RProf_{simulated} \cap RProf_{real}|/|RProf_{real}|$. A score of 0 means that both logs contain exactly the same resources, while a score of 1 indicates a complete resource divergence.

B. Experimental Results Discussion

Tables II and III show the evaluation results conducted on synthetic and real logs, respectively. Besides the probabilistic method (labeled as Prob.), we evaluated two variations of the crisp approach. Crisp calendars filter granules, group, and remove resources whose data in the event log does not meet the pre-set parameters for confidence, support, and participation [12]. To avoid resource clustering, the variant labeled naive (N-Crisp) corresponds to a configuration of these parameters set to 0. Meanwhile, the full version (C-Crisp) aligns with the optimal hyperparameters, which could lead to resource groupings. As for the C-Crisp and Prob methods, we

TABLE II: Results of the metrics on the synthetic logs.

		B-24	**U-24**	**B-8**	**U-8**	**B-8/4**	**U-8/4**	**B-24/A**	**U-24/A**
RED	N-Crisp	1.98	2.39	181.71	200.76	249.74	819.42	233.61	604.76
	C-Crisp	1.42	1.91	166.67	158.3	205.87	731.06	197.03	452.15
	Prob.	7.03	7.82	158.23	95.58	69.56	186.43	111.18	361.13
CTD	N-Crisp	2.27	2.63	436.21	445.28	414.48	1100.04	335.42	858.99
	C-Crisp	2.70	2.16	383.67	383.58	328.26	924.27	259.45	564.9
	Prob.	13.14	16.8	356.56	164.96	149.43	345.05	201.23	536.1
MMR	N-Crisp	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	C-Crisp	0.0	0.93	0.24	0.93	0.85	0.17	0.78	0.78
	Prob.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TABLE III: Results of the metrics on the real logs.

		BPIC12	**BPIC17**	**AC-CRE**	**CALL**
RED	N-Crisp	213.42	209.86	85.46	0.63
	C-Crisp	192.64	199.4	66.05	0.03
	Prob.	187.28	162.74	70.37	0.83
CTD	N-Crisp	193.01	284.56	135.49	0.97
	C-Crisp	175.1	271.88	158.11	0.04
	Prob.	170.55	227.62	97.5	1.34
MMR	N-Crisp	0.05	0.01	0.69	0.12
	C-Crisp	1.0	0.55	0.95	0.99
	Prob.	0.05	0.01	0.68	0.13

compare the results corresponding to the optimal hyperparameters that minimize the RES metric. These were determined using 30 iterations (i.e., the default value recommended by the Python library bayes_opt[9]) of a Bayesian hyper-parameter optimizer [18]. Additionally, to mitigate the impact of the simulations' stochastic nature on the results, we executed five simulations in each iteration, excluding the lowest and highest values and retaining the mean value of the remaining three. Tables II and III highlight the best approach, i.e., the most favorable (lower) results in each of the 12 logs assessed according to the RED and CTD metrics.

Regarding the synthetic evaluation, Table II shows that the Probabilistic approach tends to provide more precise results concerning the RED and CTD metrics across most of the tested configurations. More specifically, the probabilistic model overshadows the crisp one in the U-8, B-8/4, and U-8/4 in both RED and CTD metrics. In the U-8, the resources work full-time, but they have uneven workloads, better captured probabilistically than in a crisp way. In the B-8/4 and U-8/4, half of the resources work part-time, leading to lower frequencies that are more challenging to capture with a crisp model. The combination of part-time and uneven workloads in the U-8/4 spotlight the most significant differences, illustrating a scenario in which probabilistic calendars might be more suitable. Although probabilistic models perform better in B-24/A and U-24/A, the differences are less significant. The latest illustrates a limitation of both models, i.e., they are cyclical (weekly) calendars, and here, some resources are not following a cyclical schedule. Thus, the probabilistic models better capture the different resource frequencies, but an extra dimension to capture seasonal (acyclic) behavior is missing.

In Table II, the N-Crisp approach has superior accuracy only in B-24 and U-24 scenarios, which are highly crisp schedules

[9]https://github.com/bayesian-optimization/BayesianOptimization

due to the always available resources (24/7). The probabilistic approach underperforms here because it only assigns a 100% probability to a time slot if every resource starts an activity instance in every single granule in its calendar, every day, which is improbable. The crisp approach, in contrasts, assigns a 100% availability to evert granule that achieves the required confidence and support levels. Finally, in 5 out of 8 logs, crisp calendars reached MMR ratios over 0.7, indicating that replicating time accuracy comes at the expense of grouping resources, retaining only 30% or less. The latest could challenge the (specific) resource behaviors analysis and detection of resource-related issues. Conversely, probabilistic calendars retain all original resources while offering comparable or superior time estimation accuracy.

Table III also highlights a superior accuracy of the probabilistic approach in the assessed real-life logs, with the lowest RED in BPIC12 and BPIC17 and the lowest CTD in all the logs, except the CALL log where C-Crisp outperforms. In the AC-CRE log, where all approaches show MMR ratios above 0.68, this suggests that only 32% of resources in the training log appeared in the testing log, potentially introducing noise. Although C-Crisp approximates RED more closely in this scenario, the probabilistic approach achieves more substantial CTD differences, suggesting a better overall performance. The CALL log is the only dataset where the Crisp calendars outperform the Probabilistic ones among the real-life logs evaluated regarding the RED and CED metrics. Although the Probabilistic model results are close to 1, indicating acceptable accuracy, the Crisp models' lower scores suggest that the events in the CALL log are of a crisp scheduling nature. However, the high MMR values across all the C-Crisp models highlight a trade-off between temporal prediction accuracy and reproducing the exact composition of resources, a limitation absent in the probabilistic and N-Crisp models, which only excluded resources absent in the training datasets.

Threats to validity. The findings are subject to the following threats: (1) *internal validity*: the experiments rely on 8 synthetic and 4 real-life logs. The results could differ on other logs. We selected logs with varying characteristics and from different domains to mitigate this limitation. (2) *ecological validity*: we compare the simulation outputs against the original log. This allows us to measure how well the simulation models replicate the as-is process, but it does not allow us to assess the accuracy improvements of using probabilistic calendars in a what-if setting, i.e., predicting performance after a change.

VI. Conclusion

This paper introduced a process simulation approach that models resource availability via probabilistic calendars, alongside an approach to discover such calendars from event logs. An evaluation shows that simulation models with probabilistic calendars discovered from event logs more closely replicate the temporal distribution of activity instances and cases, relative to otherwise equivalent simulation models with crisp calendars. The proposed approach assumes that the periodicity of time-slots follows circadian cycles (i.e., weekly, daily, and hourly periodicity). In practical scenarios, a resource's availability might fluctuate along circannual (seasonal) cycles, e.g., availability differing between summer and winter or even within monthly cycles (e.g., availability in July is different from August). Thus, a future research direction is extending the technique to capture seasonal and even non-periodical availability patterns. The presented method additionally assumes that resources perform only one task at a time during the simulation. Another avenue for future work is the probabilistic modeling of resource multi-tasking, for example by learning models that estimate the probability of a resource taking on a new task given their current workload and availability calendar.

Reproducibility. The source code, datasets, models, and instructions to reproduce the experiments can be found at: https://github.com/orlenyslp/probabilistic_resource_calendars.

References

[1] Object Management Group, "Business Process Model and Notation (BPMN), Version 2.0.2," 2013. [Online]. Available: http://www.omg.org/spec/BPMN/2.0.2/

[2] W. M. P. van der Aalst, "Business process simulation survival guide," in *Handbook on Business Process Management 1, 2nd Ed*, 2015, pp. 337–370.

[3] Workflow Management Coalition, "BPSim: Business Process Simulation Specification," Document Number WFMC-BPSWG-2016-1, 2016, https://www.bpsim.org/specifications/2.0/WFMC-BPSWG-2016-01.pdf.

[4] W. M. P. van der Aalst, J. Nakatumba-Nabende, A. Rozinat, and N. Russell, "Business process simulation: How to get it right?" in *Handbook on Business Process Management 1*, 2010, pp. 313–338.

[5] A. P. Freitas and J. L. M. Pereira, "Process simulation support in bpm tools: The case of bpmn," 2015.

[6] M. D. Rossetti, *Simulation modeling and Arena*. Wiley, 2015.

[7] A. Rozinat, R. S. Mans, M. Song, and W. van der Aalst, "Discovering simulation models," *Inf. Syst.*, vol. 34, no. 3, pp. 305–327, 2009.

[8] N. Martin, B. Depaire, and A. Caris, "The use of process mining in business process simulation model construction - structuring the field," *Bus. Inf. Syst. Eng.*, vol. 58, no. 1, pp. 73–87, 2016.

[9] M. Camargo, M. Dumas, and O. González, "Automated discovery of business process simulation models from event logs," *Decis. Support Syst.*, vol. 134, p. 113284, 2020.

[10] N. Martin, B. Depaire, A. Caris, and D. Schepers, "Retrieving the resource availability calendars of a process from an event log," *Inf. Syst.*, vol. 88, 2020.

[11] B. Estrada-Torres, M. Camargo, M. Dumas, L. García-Bañuelos, I. Mahdy, and M. Yerokhin, "Discovering business process simulation models in the presence of multitasking and availability constraints," *Data Knowl. Eng.*, vol. 134, p. 101897, 2021.

[12] O. López-Pintado and M. Dumas, "Business process simulation with differentiated resources: Does it make a difference?" in *BPM 2022*. Springer, 2022, pp. 361–378.

[13] O. López-Pintado, I. Halenok, and M. Dumas, "Prosimos: Discovering and simulating business processes with differentiated resources," in *EDOC 2022 Workshops*. Springer, 2022, pp. 346–352.

[14] W. Lee and S. Lee, "Fuzzy calendar algebra and its applications to data mining," in *(TIME 2004)*. IEEE Computer Society, 2004, pp. 71–78.

[15] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, 2008.

[16] D. Chapela-Campa, I. Benchekroun, O. Baron, M. Dumas, D. Krass, and A. Senderovich, "Can i trust my simulation model? measuring the quality of business process simulation models," in *BPM 2023*. Springer, 2023.

[17] E. Levina and P. J. Bickel, "The earth mover's distance is the mallows distance: Some insights from statistics," in *ICCV 2001*. IEEE Computer Society, 2001, pp. 251–256.

[18] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *NIPS 2012*, 2012, pp. 2960–2968.