On the Use of Allen's Interval Algebra in the Coordination of Resource Consumption by Transactional Business Processes

Zakaria Maamar[1][a], Fadwa Yahya[2,3][b] and Lassaad Ben Ammar[2,3][c]

[1]*Zayed University, Dubai, U.A.E.*

[2]*Prince Sattam Bin Abdulaziz University, Al kharj, K.S.A.*

[3]*University of Sfax, Sfax, Tunisia*

Keywords: Allen's Interval Algebra, Business Process, Coordination, Resource, Transaction.

Abstract: This paper presents an approach to coordinate the consumption of resources by transactional business processes. Resources are associated with consumption properties known as unlimited, limited, limited-but-extensible, shareable, and non-shareable restricting their availabilities at consumption-time. And, processes are associated with transactional properties known as pivot, retriable, and compensatable restricting their execution outcomes in term of either success or failure. To consider the intrinsic characteristics of both consumption properties and transactional properties when coordinating resource consumption by processes, the approach adopts Allen's interval algebra through different time-interval relations like before, overlaps, and during to set up the coordination, which should lead to a free-of-conflict consumption. A system demonstrating the technical doability of the approach based on a case study about loan application business-process and a real dataset is presented in the paper, as well.

1 INTRODUCTION

It is known that all organizations, whether private or public, multinational or local, etc., rely on Business Processes (BP) to achieve their goals, sustain their competitiveness, improve their performance, etc. According to Weske, a BP "*is a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal*" (Weske, 2012). A BP has a process model that is designed using dedicated languages like Business Process Model and Notation (BPMN) (OMG,) and then, executed on top of a Business Process Management System (BPMS). Prior to executing BPs, their owners could have a "say" on the expected outcomes. For instance, an owner insists that her BP must succeed regardless of technical obstacles, another accepts that her BP could fail, while another approves the undoing of her BP despite the successful execution. All these options from which BPs' owners can select are framed thanks to a set of transactional properties referred to as pivot, retriable, and compensatable (Little, 2003).

On top of identifying who does what, where, when, and why, a BP's process model also identifies the necessary resources that a BP is expected to consume at run-time. These resources are of different types including human capital, software, hardware, and money. Contrarily to what some disciplines assume that resources (whether logical or physical) are abundant, we argue the opposite. It is common that resources are limited (e.g., 2 hours to complete a transaction), limited-but-extensible (e.g., 2-week validity for an access permit that can be renewed for another week), and not-shareable (e.g., a delivery truck is booked between 8am and 9am). In a previous work, we captured resources' characteristics with a set of consumption properties referred to as unlimited, limited, limited-but-extensible, shareable, and non-shareable (Maamar et al., 2016). In this paper, we examine from a temporal perspective the impact of consumption properties on transactional properties by raising questions like could a limited resource accommodate a retriable BP knowing that this resource could become unavailable after a certain number of necessary execution retrials of this BP? And, could a limited-but-extensible resource accommodate a compensatable BP knowing that extending this resource

[a] https://orcid.org/0000-0003-4462-8337
[b] https://orcid.org/0000-0003-4661-1344
[c] https://orcid.org/0000-0002-4698-3693

Maamar, Z., Yahya, F. and Ben Ammar, L.
On the Use of Allen's Interval Algebra in the Coordination of Resource Consumption by Transactional Business Processes.
DOI: 10.5220/0010887600003176
In *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2022)*, pages 15-25
ISBN: 978-989-758-568-5; ISSN: 2184-4895

would be required to support the undoing of this BP?

To address these questions, we proceed as follows. First, we blend time with consumption properties allowing to define a resource's availability-time interval. Second, we blend time with transactional properties allowing to define a BP's consumption-time interval. Finally, we resort to Allen's interval algebra, (Allen, 1983), to examine the potential relations between availability-time interval and consumption-time interval. The choice of this algebra is motivated by its exhaustive coverage of the possible relations between time intervals along with the possibility of reasoning over these relations. Examples of Allen's time relations include equals, overlaps, starts, and during. Our objective is to recommend the actions to take when a BP's consumption-time interval overlaps with a resource's availability-time interval, when a BP's consumption-time interval is during a resource's availability-time interval, when a BP's consumption-time interval and a resource's availability-time interval start or finish at the same time, etc. In a nutshell, could a BP consume a resource considering both the BP's temporal-transactional properties and the resource's temporal-consumption properties?

Our contributions are, but not limited to, (*i*) temporal analysis of consumption and transactional properties, (*ii*) illustration of how resources' availability times are adjusted to accommodate BPs' transactional properties, (*iii*) identification of Allen's relevant time relations between consumption-time and availability-time intervals, and (*iv*) development of a system allowing the reasoning over the identified Allen's relevant time relations. The rest of this paper is organized as follows. Section 2 is an overview of consumption properties, transactional properties, and Allen's interval algebra and presents some related works. Section 3 suggests a case study. Section 4 discusses the temporal coordination of processes consuming resources. Implementation details and concluding remarks are included in Sections 5 and 6, respectively.

2 BACKGROUND

We discuss consumption properties of resources, transactional properties of tasks, and Allen's interval algebra, and then, present some related works.

2.1 Resources' Consumption Properties

In compliance with our previous work on social coordination of BPs (Maamar et al., 2016), the consumption properties of a resource ($\mathcal{R}$) could be unlimited (u), shareable (s), limited (l), limited-but-extensible (lx), and non-shareable (ns). A resource is limited when its consumption is restricted to an agreed-upon period of time (capability, too, but not considered). A resource is limited-but-extensible when its consumption continues to happen after extending the (initial) agreed-upon period of time. Finally, a resource is non-shareable when its concurrent consumption needs to be coordinated (e.g., one at a time). A resource is by default unlimited and/or shareable. The consumption cycles (cc) of the 5 properties are captured into Fig. 1. However, only 2 are listed below due to lack of space.

$\mathcal{R}.cc_{ul}$: not-made-available $\xrightarrow{start}$ made available $\xrightarrow{waiting-to-be-bound}$ not-consumed $\xrightarrow{consumption-approval}$ consumed $\xrightarrow{no-longer-useful}$ withdrawn.

$\mathcal{R}.cc_{lx}$: not-made-available $\xrightarrow{start}$ made available $\xrightarrow{waiting-to-be-bound}$ not-consumed $\xrightarrow{consumption-approval}$ consumed $\xrightarrow{consumption-update}$ done $\xrightarrow{renewable-approval}$ made available. The transition from done to made available allows a resource to be regenerated for another cycle of consumption.

2.2 Tasks' Transactional Properties

The following definitions of transactional properties of a BP's tasks are reported in the literature, for instance (Frank and Ulslev Pedersen, 2012) and (Little, 2003). A task ($\mathcal{T}$) is pivot (p) when the outcomes of its successful execution remain unchanged forever and cannot be semantically undone. Should this execution fail, then it would not be retried. A task is compensatable (c) when the outcomes of its successful execution can be semantically undone. Like pivot, should this execution fail, then it would not be retried. Finally, a task is retriable (r) when its successful execution is guaranteed to happen after several finite activations. It happens that a task is both compensatable and retriable. The transactional cycles (tc) of the 3 properties are listed below:

$\mathcal{T}.tc_p^1$: not-activated $\xrightarrow{start}$ activated $\xrightarrow{commitment}$ done or $\mathcal{T}.tc_p^2$: not-activated $\xrightarrow{start}$ activated $\xrightarrow{failure}$ failed.

$\mathcal{T}.tc_r^1$: not-activated $\xrightarrow{start}$ activated 0[$\xrightarrow{failure}$ failed $\xrightarrow{retrial}$ activated]* $\xrightarrow{commitment}$ done.

$\mathcal{T}.tc_c^1$: not-activated $\xrightarrow{start}$ activated $\xrightarrow{commitment}$ done, $\mathcal{T}.tc_c^2$: not-activated $\xrightarrow{start}$ activated $\xrightarrow{failure}$ failed, or $\mathcal{T}.tc_c^3$: not-activated $\xrightarrow{start}$ activated $\xrightarrow{commitment}$ done $\xrightarrow{compensation}$ compensated.

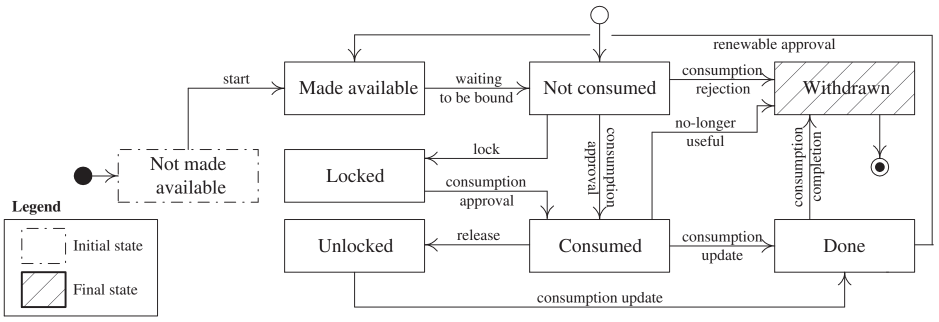


Figure 1: Representation of a resource’s consumption cycles (from (Maamar et al., 2016)).

2.3 Allen’s Interval Algebra

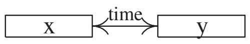
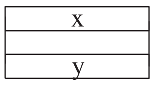
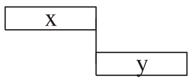
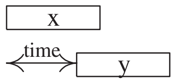
Table 1 presents some potential relations (in fact, there exist 13) between time intervals, i.e., pairs of endpoints, allowing to support multiple forms of temporal reasoning in terms of what to do when 2 time intervals start/end together, when a time interval falls into another time interval, etc. (Allen, 1983).

2.4 Related Work

Resource allocation to tasks has been dealt with using various techniques such as data mining, probabilistic allocation, and even manual allocation. Recently, process mining is also used to allocate resources to tasks. In this section, we restrict ourselves to works that adopt time-centric techniques and process mining techniques to address resource allocation concern.

In (Michael et al., 2015), Michael et al. propose a process mining-based framework for resource allocation. The framework recommends resources at the sub-process level instead of task-level by considering criteria and historical information extracted from event logs. To this end, the framework uses the best position algorithm to combine different criteria. In (Renuka et al., 2016), Sindhgatta et al. propose an approach for making decisions about resource allocation to tasks. The approach uses event log to extract information about the process context and performance from past executions. These information are analyzed using data mining techniques to discover past resource allocation decisions. The knowledge about past decisions are used by the approach to improve resource allocations in new process instances. In (Weidong et al., 2016), Zhao et al. consider resource allocation to tasks as a multi-criteria decision problem. They propose an entropy-based clustering ensemble approach for allocating resources to BPs. Recommending resources to tasks depends on tasks’ requirements and preference patterns established based on past executions. In addition, the authors adopt a heuristic technique to support dynamic resource allocation in the context of multiple process instances running concurrently. In a recent work (Zhao et al., 2020), the same authors address the problem of human resource allocation using team faultlines. They describe resources’ characteristics from demographic and past execution perspectives. They also use clustering to identify and measure team faultlines with multiple subgroups of resources and different characteristics. On top of the identified team faultlines, Zhao et al. adopt neural network for the allocation human resources to tasks. In (Rania Ben et al., 2020), Ben Halima et al. propose an approach that ensures time-aware allocation of cloud resources to BPs. The approach prevents violating time constraints on the processes while minimizing the deployment cost of these processes. First, the approach uses timed automata for a formal verification of the matching between BPs’ temporal constraints and cloud resources’ time availabilities. Then, linear programming is used to optimize costs. In (Alessandro et al., 2020), Stefanini et al. suggest a process mining-based approach to support the planning of resources in healthcare services. The approach allows a semi-automatic extraction and evaluation of the tasks, service times, and resource consumption for specific medical conditions from the event logs. Indeed, the approach estimates the expected resource consumption for a pre-defined period. In (Delcoucq et al., 2020), Delcoucq et al. consider that process mining techniques are mainly used to address control-flow related problems like BP’s performance or compliance. However, limited works exist on using these techniques to address resource allocation. Our proposal is a step in this direction as we adopt Allen’s interval algebra to assign availability intervals to resources and consumption intervals to tasks. We also adopt process mining to allocate resources to tasks based on their respective availability and consumption intervals.

Table 1: Representation of some Allen's time-interval relations.

			
x before y	*x equals y*	*x meets y*	*x overlaps y*

3 CASE STUDY

To illustrate task-resource coordination from a time perspective, we adapt the case study of loan-application BP that was used in the context of BPI challenge 2017 (van Dongen, 2017).

The process model of the loan-application BP begins when a customer submits an online application that a credit staff checks for completeness. Should any documentation be missing, the staff would contact the customer prior to processing the application further. Otherwise, the staff would do some additional work like assessing the customer's eligibility based on the requested amount, income, and history. Should the customer be eligible, the staff would prepare a proposal that the customer has to either accept or reject according to a deadline. After this deadline and in the absence of a response, the application is automatically cancelled. Otherwise, the staff would finalize the necessary paperwork by seeking the manager's approval. Finally, the customer would be informed of the approval concluding the whole process.

For our needs, we split the loan-application BP into 3 parts, review application, prepare proposal, and make decision, that are subject to temporal constraints that, if not met, would terminate the application. To complete the review-application part, the customer has to submit any missing documentation *during* a certain time period. This period can be *extended* according to how far the credit staff is from her monthly target of loan applications to process.

To complete the prepare-proposal part, the credit staff checks the customer's eligibility, waits for some additional details like credit score that the central bank provisions, and finally prepares the proposal. All this needs to happen *within* 15 days from the submission date as instructed by the central bank.

To complete the make-decision part, the staff must receive the customer's response to the proposal and seek the manager's approval *before* the 15 days limit. Otherwise, the application is cancelled.

Based on the description above, we identify some concerns that task/resource coordination is expected to address. Firstly, ensuring resource availability impacts the outcomes of some tasks. For instance, the customer's response to the bank's proposal must be received while the credit staff is on duty and within the 15 days limit. Should the staff be off duty, then analysing the response should be allocated to a different staff while satisfying this limit as well. Secondly, extending resource availability permits to accommodate the execution of more tasks instead of cancelling them. For instance, securing customers' approvals about any potential delays would help the bank remain compliant with the 15 days limit. Finally, providing an efficient resource allocation plan would improve the process performance.

4 COORDINATION OF TASKS CONSUMING RESOURCES

This section details our approach for coordinating task/resource consumption from a time perspective (Fig. 2). The approach goes through 3 stages though the first 2 happen concurrently. In the first stage, the approach blends time with consumption properties allowing to define the availability-time interval of a resource (Section 4.1). In the second stage, the approach blends time with transactional properties allowing to define the consumption-time interval of a task (Section 4.2). Finally, the approach examines the overlap between availability-time interval and consumption-time interval according to Allen's interval algebra (Section 4.3). This overlap corresponds to the coordination that should take place when tasks consume resources. The approach also mines processes by analysing logs to guide task and resource coordination. More details about Fig. 2's modules, repositories, and operations are given in Section 5.

4.1 Time/Consumption Properties Blend

To decide when a task ($\mathcal{T}_i$) would consume ($con_{i\{j=1...\}}$, one-to-many times) a resource ($\mathcal{R}_k$), we proceed as follows. First, we associate the effective consumption with a time interval, $\mathcal{T}_i^{\mathcal{R}_k}[x_{con_{ij}}, y_{con_{ij}}]$, that will be defined at run-time (Section 4.3). Second, we associate unlimited, limited, limited-but-extensible properties with

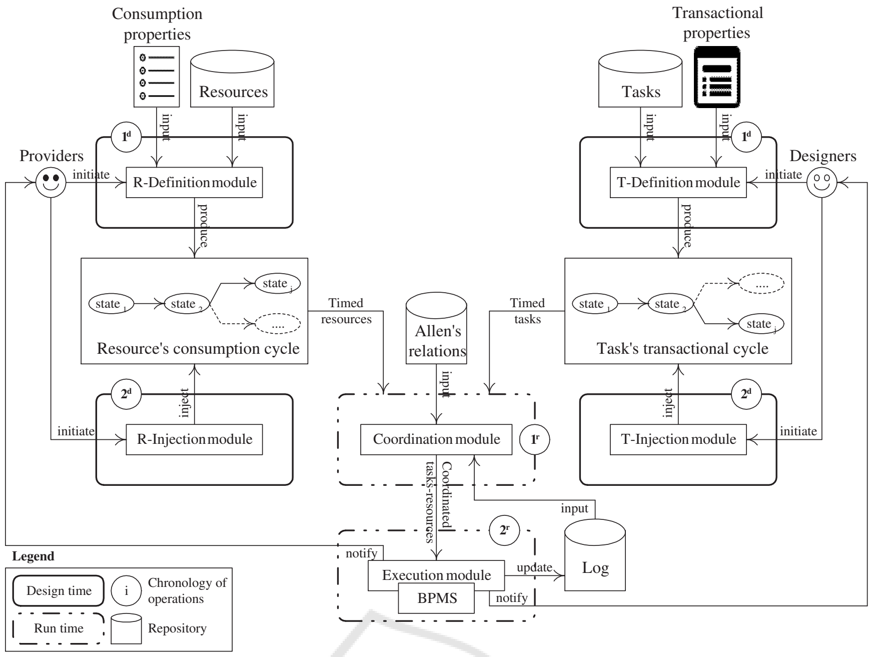


Figure 2: Approach for coordinating tasks consuming resources from a time perspective.

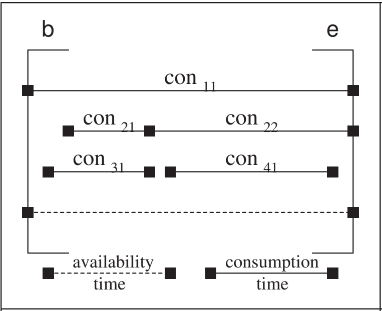
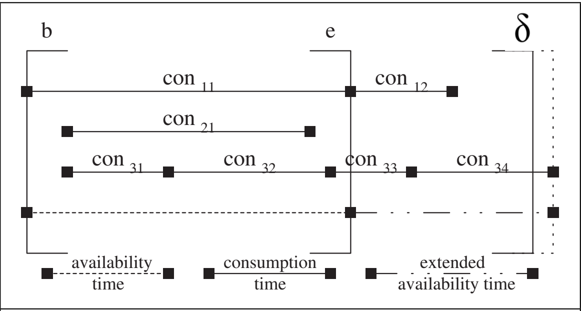
3 intervals defining a resource's availability time, $\mathcal{R}_k[b,e[$, $\mathcal{R}_k[b,e]$, $\mathcal{R}_k[b,e\ 0[+\delta]*]$, where b, e, δ, and $0[\ldots]*$ stand for begin-time, end-time, extra-time[1], and 0-to-many times. Third, we associate shareable and non-shareable properties with tolerating the concurrent consumption $(con_{in}, con_{jn'}, \ldots)$ of a resource by separate tasks $(\mathcal{T}_i, \mathcal{T}_j, \ldots)$ during the availability time of this resource, e.g., $((\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{in}}, e_{con_{in}}] \subseteq \mathcal{R}_k[b,e]) \wedge (\mathcal{T}_j^{\mathcal{R}_k}[b_{con_{jn'}}, e_{con_{jn'}}] \subseteq \mathcal{R}_k[b,e]) \wedge \ldots)$ where $i, j \in 1..N$, N is the number of tasks, and $n, n' \in \mathbb{N}^*$. Finally, we ensure that an unlimited resource accommodates any task's multiple consumption requests. Using Table 2 that refers to $\mathcal{T}_{1,2,3}$ and their different resource consumption such as $\mathcal{T}_1$'s con_{11} and $\mathcal{T}_3$'s $con_{31,32,33,34}$, we discuss the impact of limited and limited-but-extensible properties on a resource's availability-time interval.

1. Limited property means that a resource's availability time, $\mathcal{R}_k[b,e]$, set at design-time, remains the same at run-time despite the additional resource consumption coming from the same tasks (after their first consumption). A task requesting to consume a limited resource is confirmed *iff* the task's first consumption-time falls into the resource's availability time (e.g., $\mathcal{T}_2^{\mathcal{R}_k}[b_{con_{21}}, e_{con_{21}}] \subset \mathcal{R}_k[b,e]$ in Table 2 (a) where $b_{con_{21}} > b$ and $e_{con_{21}} < e$) and, then, any extra consumption times must fall into the resource's same availability time (e.g., $\mathcal{T}_2^{\mathcal{R}_k}[b_{con_{22}}, e_{con_{22}}] \subseteq r_k[b,e]$ in Table 2 (a) where $b_{con_{22}} = e_{con_{21}}$ and $e_{con_{22}} = e$).
2. Limited-but-extensible property means that a resource's availability time, $\mathcal{R}_k[b,e]$, set at design-time, can be adjusted at run-time, $\mathcal{R}_k[b,e\ 0[+\delta]*]$, so, that, additional resource consumption coming from the same tasks (after their first consumption) are accommodated. A task requesting to use a limited-but-extensible resource is confirmed *iff* the task's first consumption-time falls into the resource's availability time (e.g., $\mathcal{T}_3^{\mathcal{R}_k}[b_{con_{31}}, e_{con_{31}}] \subset \mathcal{R}_k[b,e]$ in Table 2 (b) where $b_{con_{31}} > b$ and $e_{con_{31}} < e$) and, then, any extra consumption times still fall into either the resource's same availability time (e.g., $\mathcal{T}_3^{\mathcal{R}_k}[b_{con_{32}}, e_{con_{32}}] \subset \mathcal{R}_k[b,e]$ in Table 2 (b) where $b_{con_{32}} = e_{con_{31}}$ and $e_{con_{32}} < e$) or the resource's extended availability time (e.g., $\mathcal{T}_3^{\mathcal{R}_k}[b_{con_{33}}, e_{con_{33}}] \subset \mathcal{R}_k[e, e+\delta]$ in Table 2 (b) where $b_{con_{33}} = e_{con_{32}}$ and $e_{con_{33}} < e+\delta$).

In the 2 cases above, we assume that any additional resource consumption happens immediately after the end of the previous resource consumption, i.e., $b_{con_{ij}} = e_{con_{ij+1}}$. Another option is to have a gap (γ) between the 2 consumption, i.e., $b_{con_{ij}} = e_{con_{ij+1}} + \gamma$, but this is not considered further and does not impact the whole coordination approach.

[1]Could be repeated but not indefinitely.

Table 2: Representation of a resource's availability-time intervals during consumption.

	
(a) limited resource	(b) limited-but-extensible resource

4.2 Time/Transactional Properties Blend

In Section 4.1, we refer to $\mathcal{T}_i^{\mathcal{R}_k}[x_{con_{ij}}, y_{con_{ij}}]$ as a task's effective consumption-time interval with regard to a resource. This interval will be defined based on a task's expected consumption-time interval, $\mathcal{T}_i[et, lt]$, where *et* and *lt* are earliest time and latest time, respectively. While the expected consumption time is set at design-time, the effective consumption time will be set at run-time as per Section 4.3 and will happen anytime between the earliest time and latest time. We discuss below how we foresee the impact of a task's transactional properties on a resource's availability-time intervals.

1. Pivot: a resource's availability-time interval accommodates the execution of a task whether this execution leads to success or failure.
2. Compensatable: a resource's availability-time interval accommodates the execution of a task whether this execution leads to success or failure. Prior to undoing the execution outcomes after success (assuming an undoing decision has been made), there will be a need to check whether the resource's remaining availability time accommodates the undoing. Should the accommodation be not possible, extra availability time would be requested subject to checking the resource's consumption property.
3. Retriable: a resource's availability-time interval accommodates the execution of a task along with an agreed-upon number of retrials, if deemed necessary, that are all expected to lead to success. Should this number of retrials still lead to failure, extra availability time would be requested subject to both checking the resource's consumption property and ensuring that the extra number of retrials (that are expected to lead to success) do not go over a threshold.

4.3 Task/Resource Time-connection

To define the effective consumption-time interval of a resource by a task, we examine potential overlaps between the task's expected consumption-time interval and resource's availability-time interval. We resort to Allen's interval algebra to identify these overlaps, i.e., $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{in}}, e_{con_{in}}]$ where n is the consumption number. In the following, $\mathcal{TH}$, $\mathcal{R}_k(b|e)$, and $\mathcal{T}_i(et|lt)$ correspond to threshold during retriable execution, lower|upper values of a resource's availability-time interval, and lower|upper values of a task's excepted consumption-time interval, respectively. Due to lack of space, equals and overlaps relations are detailed and during relation is summarized.

Consumption-time Interval *Equals* Availability-time Interval. Since the expected consumption-time interval and availability-time interval are the same, the following would happen considering both the resource's consumption property and the task's transactional property:

1. limited: the effective consumption-time interval, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$, falls into the availability-time interval in a way that $b_{con_{i1}} \geq \mathcal{R}_k(b)$ and $e_{con_{i1}} \leq \mathcal{R}_k(e)$.
   - pivot: $\mathcal{T}_i$ execution happens during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$.
   - compensatable: $\mathcal{T}_i$ execution happens during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$. Should there be an undoing of this execution's outcomes, then the remaining availability time would accommodate the undoing, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i2}}, e_{con_{i2}}]$ and $b_{con_{i2}} = e_{con_{i1}}$, subject to verifying that $\mathcal{R}_k(e) - e_{con_{i1}} > 0$. Should the verification fail, then the compensation would be canceled. Despite the cancellation, $\mathcal{T}_i$ execution is still compliant with the requirements of a compensatable property (i.e., done is a final state).
   - retriable: $\mathcal{T}_i$ execution and potential agreed-

upon retrials happen during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$. Should there be extra retrials to ensure successful execution, then the remaining availability time would accommodate these extra retrials, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{in}}, e_{con_{in}}]$ and $b_{con_{in}} = e_{con_{i(n-1)}}$, subject to verifying that $\mathcal{R}_k(e) - e_{con_{i(n-1)}} > 0$ where $1 < n < \mathcal{TH}$. Should the verification fail, then the retrials would be stopped making $\mathcal{T}_i$ execution uncompliant with the requirements of a retriable property (i.e., failed is not a final state).

2. limited-but-extensible: the effective consumption-time interval, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$, falls into the availability-time interval in a way that $b_{con_{i1}} \geq \mathcal{R}_k(b)$ and $e_{con_{i1}} \leq \mathcal{R}_k(e)$.
   - pivot: $\mathcal{T}_i$ execution happens during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$.
   - compensatable: $\mathcal{T}_i$ execution happens during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$. Should there be an undoing of this execution's outcomes, then the remaining availability time would accommodate the undoing, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i2}}, e_{con_{i2}}]$ and $b_{con_{i2}} = e_{con_{i1}}$, subject to verifying that $\mathcal{R}_k(e) - e_{con_{i1}} > 0$. Should the verification fail, then the resource's availability time would be extended, $\mathcal{R}_k(e) + \delta$, in a way that $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i2}}, e_{con_{i2}}] \subseteq \mathcal{R}_k[e_{con_{i1}}, e + \delta]$, $b_{con_{i2}} = e_{con_{i1}}$, and $e_{con_{i2}} \leq \mathcal{R}_k(e) + \delta$. Whether the extension happens or not, $\mathcal{T}_i$ execution is still compliant with the requirements of a compensatable property (i.e., both done and canceled are final states).
   - retriable: $\mathcal{T}_i$ execution and potential agreed-upon retrials happen during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$. Should there be extra retrials to ensure successful execution, then the remaining availability time would accommodate these extra retrials, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{in}}, e_{con_{in}}]$ and $b_{con_{in}} = e_{con_{i(n-1)}}$, subject to verifying that $\mathcal{R}_k(e) - e_{con_{i(n-1)}} > 0$ where $1 < n < \mathcal{TH}$. Should the verification fail, then the resource's availability time would be extended a certain number of times, $\mathcal{R}_k(e) + 1[\delta]*$, in a way that $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{in}}, e_{con_{in}}] \subseteq \mathcal{R}_k[e_{con_{i(n-1)}}, e + 1[\delta]*]$, $b_{con_{in}} = e_{con_{i(n-1)}}$, $e_{con_{in}} \leq \mathcal{R}_k(e) + \delta$, and $1 < n < \mathcal{TH}$. Thanks to the extension, $\mathcal{T}_i$ execution is still compliant with the requirements of a retriable property (i.e., done is a final state).

Consumption-time Interval *Overlaps* Availability-time Interval. Since the expected consumption-time interval and availability-time interval have some time in common, the lower value of the task's expected consumption-time interval is adjusted in a way that it matches the lower value of the resource's availability-time interval, i.e., $\mathcal{T}_i = [\mathcal{R}_k(b), lt]$. After this adjustment, the following would happen considering both the resource's consumption property and the task's transactional property:

1. limited: the effective consumption-time interval, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$, falls into a part of the availability-time interval in a way that $b_{con_{i1}} \geq \mathcal{R}_k(b)$ and $e_{con_{i1}} < \mathcal{R}_k(e)$. The remaining part of the availability-time interval, that corresponds to $[e_{con_{i1}}, (\mathcal{R}_k(e) - e_{con_{i1}})]$, could be used to accommodate additional consumption (Table 2-(a)).
   - pivot: $\mathcal{T}_i$ execution happens during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$.
   - compensatable: $\mathcal{T}_i$ execution happens during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$. Should there be an undoing of this execution's outcomes, then the remaining availability time would accommodate the undoing, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i2}}, e_{con_{i2}}]$ and $b_{con_{i2}} = e_{con_{i1}}$, subject to verifying that $\mathcal{R}_k(e) - e_{con_{i1}} > 0$. Should the verification fail, then the compensation would be canceled. Despite the cancellation, $\mathcal{T}_i$ execution is still compliant with the requirements of a compensatable property (i.e., done is a final state).
   - retriable: $\mathcal{T}_i$ execution and potential agreed-upon retrials happen during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$. Should there be extra retrials to ensure successful execution, then the remaining availability time would accommodate these extra retrials, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{in}}, e_{con_{in}}]$ and $b_{con_{in}} = e_{con_{i(n-1)}}$, subject to verifying that $\mathcal{R}_k(e) - e_{con_{i(n-1)}} > 0$ where $1 < n < \mathcal{TH}$. Should the verification fail, then the extra retrials would be stopped making $\mathcal{T}_i$ execution uncompliant with the requirements of a retriable property (i.e., failed is not a final state).

2. limited-but-extensible: the effective consumption-time interval, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$, falls into a part of the availability-time interval in a way that $b_{con_{i1}} \geq \mathcal{R}_k(b)$ and $e_{con_{i1}} < \mathcal{R}_k(e)$. The remaining part of the availability-time interval, that corresponds to $[e_{con_{i1}}, (\mathcal{R}_k(e) - e_{con_{i1}})]$, could be used to accommodate additional consumption before considering to extend this availability-time interval (Table 2-(b)).
   - pivot: $\mathcal{T}_i$ execution happens during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$.
   - compensatable: $\mathcal{T}_i$ execution happens during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$. Should there be an un-

doing of this execution's outcomes, then the remaining availability time would accommodate the undoing, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i2}}, e_{con_{i2}}]$ and $b_{con_{i2}} = e_{con_{i1}}$, subject to verifying that $\mathcal{R}_k(e) - e_{con_{i1}} > 0$. Should the verification fail, then the resource's availability time would be extended, $\mathcal{R}_k(e) + \delta$, in a way that $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i2}}, e_{con_{i2}}] \subseteq \mathcal{R}_k[e_{con_{i1}}, e + \delta]$, $b_{con_{i2}} = e_{con_{i1}}$, and $e_{con_{i2}} \leq \mathcal{R}_k(e) + \delta$. Whether the extension happens or not, $\mathcal{T}_i$ execution is still compliant with the requirements of a compensatable property (i.e., both done and canceled are final states).

- retriable: $\mathcal{T}_i$ execution and potential agreed-upon retrials happen during $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{i1}}, e_{con_{i1}}]$. Should there be extra retrials to ensure successful execution, then the remaining availability time would accommodate these extra retrials, $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{in}}, e_{con_{in}}]$ and $b_{con_{in}} = e_{con_{i(n-1)}}$, subject to verifying that $\mathcal{R}_k(e) - e_{con_{i(n-1)}} > 0$ where $1 < n < \mathcal{TH}$. Should the verification fail, then the resource's availability time would be extended a certain number of times, $\mathcal{R}_k(e) + 1[\delta]*$, in a way that $\mathcal{T}_i^{\mathcal{R}_k}[b_{con_{in}}, e_{con_{in}}] \subseteq \mathcal{R}_k[e_{con_{i(n-1)}}, e + 1[\delta]*]$, $b_{con_{in}} = e_{con_{i(n-1)}}$, $e_{con_{in}} \leq \mathcal{R}_k(e) + \delta$, and $1 < n < \mathcal{TH}$. Thanks to the extension, $\mathcal{T}_i$ execution is still compliant with the requirements of a retriable property (i.e., done is a final state).

Availability-time Interval *Overlaps* Consumption-time Interval. Although "availability-time interval overlaps consumption-time interval" looks similar to "consumption-time interval overlaps availability-time interval", there are 2 adjustments that need to take place considering the fact that availability-time interval and expected consumption-time interval have some time in common. The first adjustment consists of matching the lower value of the resource's availability-time interval with the lower value of the task's expected consumption-time interval, i.e., $\mathcal{R}_k = [b', e]$ and $b' = \mathcal{T}_i(et)$. The second adjustment is conditional since it applies to limited-but-extensible resources[2], only, and consists of matching the highest value of the resource's availability-time interval with the highest value of the task's expected consumption-time interval, i.e., $\mathcal{R}_k = [b', e']$ and $e' = \mathcal{T}_i(lt)$. Extending from the beginning permits to offer a complete coverage of tasks' expected consumption-time intervals instead of waiting for these tasks' extension requests. As a result of the second adjustment, the analysis of "availability-time overlaps consumption-time" becomes similar to "availability-time equals consumption-time" after dropping limited resources from the analysis.

Availability-time Interval *during* Consumption-time Interval. 2 adjustments need to happen as follows. The first adjustment consists of adjusting the lower value of the task's expected consumption-time to match the lower value of the resource's availability-time interval, i.e., $\mathcal{T}_i = [\mathcal{R}_k(b), lt]$. The second adjustment is applicable only if the resource is limited-but-extensible. This consists of extending the resource availability-time interval so, that, its highest value matches the highest value of the task's expected consumption-time interval, i.e., $\mathcal{R}_k = [b, e']$ and $e' = \mathcal{T}_i(lt)$. As a result of these adjustments, the analysis of "availability-time interval during consumption-time interval" becomes similar to "availability-time interval equals consumption-time interval".

5 IMPLEMENTATION

To demonstrate the technical doability of our approach for coordinating the consumption of resources by processes from a time perspective, we deployed an in-house testbed that uses BPI-Challenge-2017's real dataset (van Dongen, 2017). The dataset is about a credit application system's execution traces and is available in eXtensible Event Stream (XES). In conjunction with this dataset, we developed several Python programs and performed several experiments.

5.1 Dataset Preprocessing

Necessary steps for preprocessing BPI-Challenge-2017's dataset are detailed below:

- Data transformation from XES into pandas DataFrame[3] format aims at tapping into pandas library's predefined routines which we applied to the next preprocessing steps. DataFrame is suitable for manipulating data and building prediction models.
- Data reduction/sampling downsized the dataset to make it manageable. We applied sampling to extract a representative set, 30%, of the original dataset. This number was chosen for performance purposes, given the dataset's initial size and time spent considering other data volumes.
- Feature selection considered a subset of the features describing relevant data. The objective is

[2]Considering limited resources could be an option. But, this would require reducing a task's expected consumption-time interval, which might not be practical in real-life.

[3]pandas.pydata.org/docs/reference/api/pandas.DataFrame.html.

to obtain a concise representation of the available data in the reduced dataset such as *org:resource*, *lifecycle:transition*, and *time:timestamp*. Feature selection also permits to drop irrelevant data such as *case:LoanGoal*, *MonthlyCost*, and *case:RequestedAmount* that could slowdown our system or affect the inaccuracy of the results.

- Feature creation generated new features that aim at capturing the most important data in a dataset much more efficiently than the original dataset. We used 2 techniques, feature extraction and feature construction. In the former technique, we extracted the resource's availability time, $\mathcal{R}_k[b,e]$, from the *time:timestamp* feature in the dataset. For each resource $\mathcal{R}_k$, *b* and *e* refer, respectively, to the lowest and highest values per date of the *time* part in the *time:timestamp* feature. In the latter technique, we performed a deep analysis of the reduced dataset to define the consumption-time interval of the resource, $\mathcal{T}_i^{\mathcal{R}_k}[x_{con_{ij}}, y_{con_{ij}}]$. For each task, we computed the time between its instances and their successors. The average time is then considered as the consumption-time interval for that task.

  Once a resource's availability-time interval and consumption-time interval are defined, we worked on task/resource time-connection as per Section 4.3. In addition, the analysis of the dataset and several papers on the BPI challenge 2017 allowed us to work on the consumption property of each resource and transactional property of each task. For instance, we noticed that the execution of some tasks was suspended when the resource availability-time ended. Hence, we assigned *limited* property to this resource. As for the transactional property, we associated *compensatable* property with *make offer* task since 3% of offers were canceled or refused after their successful execution according to (Bolt, 2017). Finally, we added the consumption and transactional properties to the preprocessed dataset.

5.2 Task/Resource Recommendations

To enact task/resource coordination, we resorted to process mining that is known for addressing BP decision-making problems and providing recommendations to improve future BP executions. In this context, a good number of recommendations techniques are reported in the literature. We opted for Decision Trees (DT)[4] known for easiness, performance, and ability to visually communicate choices. We created a prediction model following 2 stages, offline and online, allowing to build and process the required prediction model.

Offline Stage. We built a prediction model by referring to an open-source Python library called Sklearn[5]. It supports a variety of built-in prediction models (e.g., DT, KNN, and SVM). Building a prediction model with Sklearn usually starts with preparing the dataset in the most suitable format as per Section 5.1 that is Pandas DataFrame in our case. Then, the target of the prediction model must be defined. This latter is about the recommendation of the actions to take when a resource is assigned to a task (e.g., adjusting the resource-availability time or consumption-time interval). Finally, the set of variables that may affect the recommendations such as resource's availability-time and consumption-time are defined.

In terms of technical details, we, first, selected the DT model and then, set its parameters namely, attribute selection criterion (Entropy or Gini index) and maximum depth of the tree. These parameters are critical to the accuracy of the results and are usually set manually after several trials to find the best results. Afterwards, we fitted the DT model into the specified data. We had to split the reduced dataset into training dataset and test dataset using Sklearn's TRAIN_TEST_SPLIT[6] built-in function. Finally, we evaluated the accuracy of our model by comparing the result of the prediction to the real data included in the test dataset as per Section5.3.

Online Stage. The objective here is to recommend actions to take when a resource is assigned to a task in the new executions. Obviously, recommendations are determined using the prediction model built during the offline stage. We carried out some experiments to predict the actions to perform for each task-resource with respect to some new simulated BP instances. We checked how accurate our prediction model is. During the experiments, we considered 30 simulated instances of the BP. The accuracy of results are discussed next.

5.3 Discussions of the Results

To appreciate our DT-based prediction model's recommendations, we also adopted the k-Nearest Neighbors (KNN) as another technique for developing prediction models. As for the DT prediction model, the experiments showed encouraging results during either the offline stage or the online stage. In this context,

[4]scikit-learn.org/stable/modules/tree.html.

[5]scikit-learn.org/stable/index.html.

[6]scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

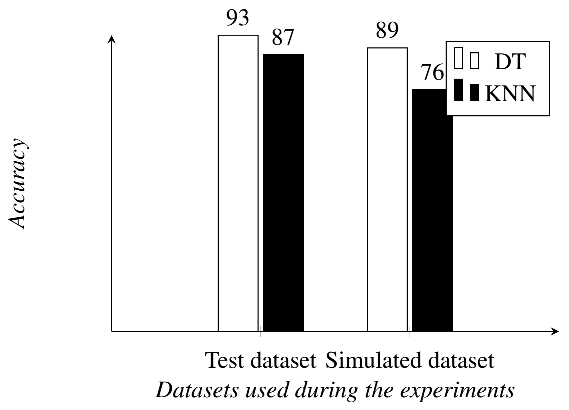


Figure 3: Accuracy of DT *versus* KNN.

we used *Accuracy* as a performance measure (Equation 1).

$$Accuracy = \frac{TrueObservations}{TotalObservations} \tag{1}$$

First, we computed *Accuracy* after applying the DT prediction model to the test dataset as per the offline stage. We obtained 93% *Accuracy*, which we treated as an acceptable value. Then, we computed *Accuracy* again after applying our prediction model during the online stage to the simulated instances. We obtained 89% *Accuracy* that also proves the high performance of our DT prediction model. Last but not least, we used KNN prediction model. Similarly to the DT prediction model, we used our preprocessed dataset to build the KNN prediction model and, then, we applied it to the simulated instances. The objective is to compare the results obtained using both techniques (DT and KNN). The results show an *Accuracy* of 87% and 76% when the KNN prediction model is applied to the test dataset and the simulated instances. These results are in line with those of the DT and proves its performance. Fig. 3 illustrates the comparison between the *Accuracy* values obtained by DT and KNN prediction models using both the test dataset and the simulated instances.

6 CONCLUSION

This paper presented an approach for coordinating the consumption of resources by business processes. The coordination took place from a time perspective thanks to specific relations defined by Allen's interval algebra. Examples of relations included equals, overlaps, starts, and finishes. The coordination also considered both the consumption properties of resources (unlimited, limited, limited-but-extensible, shareable, and non-shareable) and the transactional properties of business processes (pivot, retriable, and compensatable). Since resources and processes were associated with availability-time intervals and consumption-time intervals, respectively, different types of reasoning took place leading sometimes to adjusting some time intervals and/or allowing some re-executions. A system demonstrating the technical doability of the approach based on a case study about loan application business-process was implemented. We built a decision-tree prediction model using a real dataset from the BPI Challenge 2017. The evaluation of this prediction model proved its performance. In addition, to consolidate these results, we carried out a second experiment using KNN whose results were in line with those of DT and have proven its performance. In term of future work we would like to examine the remaining Allen's time relations, the scalability of the system when a large number of processes are under-execution, and finally, the impact of resource failure on process execution.

REFERENCES

Alessandro, S., Davide, A., Elisabetta, B., Riccardo, D., and Valeria, M. (2020). A Data-driven Methodology for Supporting Resource Planning of Health Services. *Socio-Economic Planning Sciences*, 70.

Allen, J. (1983). Maintaining Knowledge about Temporal Intervals. *CACM*, 26(11).

Bolt, A. (September 2017). BPI Challenge 2017 Process Variability Analysis over Offers. https://www.win.tue.nl/bpi/lib/exe/fetch.php?media=2017:bpi2017_paper_22.pdf,visitedMay~2021.

Delcoucq, L., Lecron, F., Fortemps, P., and van der Aalst, W. M. (2020). Resource-centric Process Mining: Clustering using Local Process Models. In *Proceedings of SAC'2020*, online.

Frank, L. and Ulslev Pedersen, R. (2012). Integrated Distributed/Mobile Logistics Management. *TLDKS*, 5.

Little, M. (2003). Transactions and Web Services. *CACM*, 46(10).

Maamar, Z., Faci, N., Sakr, S., Boukhebouze, M., and Barnawi, A. (2016). Network-based Social Coordination of Business Processes. *IS*, 58.

Michael, A., Eric, R., Jorge, M., and Marcos, S. (2015). A Framework for Recommending Resource Allocation based on Process Mining. In *Proceedings of BPM'2015 Workshops*, Innsbruck, Austria.

OMG. Business Process Model and Notation. www.omg.org/spec/BPMN/2.0.2.

Rania Ben, H., Kallel, K., Gaaloul, W., Maamar, Z., and Jmaiel, M. (2020). Toward a Correct and Optimal Time-aware Cloud Resource Allocation to Business Processes. *FGCS*, 112.

Renuka, S., Aditya, G., and Hoa Khanh, D. (2016). Context-aware Analysis of Past Process Executions to

Aid Resource Allocation Decisions. In *Proceedings of CAiSE'2016*, Ljubljana, Slovenia.

van Dongen, B. (2017). BPI Challenge 2017. data.4tu.nl/articles/dataset/BPI_Challenge_2017/12696884.

Weidong, Z., Haitao, L., Weihui, D., and Jian, M. (2016). An Entropy-based Clustering Ensemble Method to Support Resource Allocation in Business Process Management. *KIS*, 48(2).

Weske, M. (2012). Business Process Management Architectures. In *Business Process Management*. Springer.

Zhao, W., Pu, S., and Jiang, D. (2020). A Human Resource Allocation Method for Business Processes using Team Faultlines. *Applied Intelligence*, 50(9).