Interactive Design of Time-Aware Business Processes

Keti Lila[0000−0003−1634−429X], Marco Franceschetti[0000−0001−7030−282X], and Julius Köpke[0000−0002−6678−5731]

Department of Informatics-Systems, Alpen-Adria Universität Klagenfurt, Universitaetsstrasse 65-67, 9020 Klagenfurt am Wörthersee, Austria
[keti.lila|marco.franceschetti|julius.koepke]@aau.at
https://www.aau.at/isys/

Abstract. In recent years, the design of time-aware business processes has seen advancements mostly in the theory. Various modeling constructs and techniques to verify the temporal qualities of business processes have been formalized. However, operational support for process designers still lags behind, with only a few tools offering limited time-related functionalities available. Here, we contribute towards closing this gap and propose a modeling tool based on the Camunda modeler. Our tool features an interactive interface that aids the design of time-aware process models with guarantees of temporal correctness in terms of dynamic controllability.

Keywords: Process modeling tools · Time-aware processes · Dynamic controllability.

1 Introduction

The design of a business process model requires defining elements from the five process modeling aspects - functional, behavioral, operational, informational, and organizational. However, research around time-aware processes in the last two decades has shown that the temporal aspect ought to be considered as an additional aspect to tackle in the modeling phase [2].

The temporal aspect includes the definition of properties such as durations for tasks, temporal constraints, deadlines, recurring events, etc. as pointed out by extensive prior work (for a comprehensive overview, see the time patterns in [7]). Modeling the temporal aspect, however, makes process design significantly more challenging. This is due to the fact that the temporal aspect may introduce potential conflicts in a process model, which are difficult to recognize and identify for human designers. Examples of such conflicts are temporal constraints between events that cannot be satisfied at the same time or temporal constraints that may be satisfied only for certain task durations that cannot be controlled.

Checking for the absence of conflicting temporal constraints, resp. the possibility of executing a process without violating temporal constraints independent of uncontrollable durations led to the formulation of properties, the most notable of which are satisfiability and dynamic controllability (DC) [1]. In a nutshell, a

process model is satisfiable if it admits at least one trace satisfying all temporal constraints; it is dynamically controllable if, for any observed value for uncontrollable durations and conditions at xor-splits, the executor can dynamically steer the execution satisfying all constraints. Dynamic controllability is considered a highly desirable property since it allows for high flexibility yet with strong guarantees of avoidance of violations.

Typically, DC-checking procedures are applied to fully specified process models at the end of the design phase. However, it would be beneficial for process designers to receive guidance *during* the process design phase for reaching process models that are dynamically controllable, rather than finding out at the end of the design phase whether their processes are dynamically controllable. With such guidance, process designers may enjoy the additional benefit of receiving information about which are the admissible values they may set for the various temporal elements, e.g., which is the maximal allowed bound for a constraint stating the maximum time to elapse between two events that guarantees DC.

To the best of our knowledge, only TimeAwareBPMN-js [8] provides some kind of support during the design phase. However, in TimeAwareBPMN-js it is the responsibility of the designer to decide when to perform a DC-check to receive information regarding the DC property and get informed where conflicts may exist and how they may be fixed. Additionally, TimeAwareBPMN-js does not allow defining temporal parameters, which model external events [3].

Here, we propose TemporalBPMN-ID, an interactive process designer tool, which allows designers to model a business process in a BPMN-like language extended with constructs for the temporal aspect[1]. The tool proactively provides designers information about the DC property of the process being modeled. The tool provides also information about the admissible values for newly added temporal constraints before designers assign them a value. With this proactive approach, designers can focus solely on the modeling of processes, reducing the challenges induced by the temporal aspect. The proposed tool has a microservice architecture, which enables modularization, scalability, and extensibility.

2 Time-Aware Business Processes

We consider business processes to be time-aware when they include elements such as events, activity durations, and temporal constraints between events.

Events may correspond to the start and the end times for control flow elements (e.g., activities, gateways), as well as to time points that do not refer to control flow elements. Events not bound to control flow elements can be encoded through temporal variables, which can be exchanged between local processes in the form of temporal parameters [5]. Our tool supports the definition of events related to control flow elements as well as temporal parameters.

Traditionally, a distinction is made between contingent and non-contingent durations, i.e., durations that cannot be controlled but only observed (e.g., a

[1] The tool homepage is http://isys.uni-klu.ac.at/pubserv/tools/TemporalBPMN-ID/. It also contains a screen cast demonstrating the tool functionalities.

bank money transfer, guaranteed to take between 1 and 4 days) versus durations that can be controlled and decided by the process controller (e.g., writing a document) [1]. In our tool, we support the definition of both types of durations.

Temporal constraints restrict the time of occurrence of pairs of events relative to each other. Such a restriction may be either an upper-bound, specifying a maximum allowed time distance, or a lower-bound, specifying a minimum allowed time distance [4]. In our tool, we support the definition of both upper- and lower-bound constraints between events.

Dynamic controllability is among the most relevant properties for time-aware business processes. A time-aware business process is dynamically controllable if, for every possible setting of contingent durations and uncontrollable conditions at xor-splits, the controller can set the values for all subsequent non-contingent events in a way that all temporal constraints can be satisfied. DC is considered the most relaxed notion of temporal correctness that offers guarantees of no temporal constraint violations. Our tool supports the check for dynamic controllability. In order to perform a DC-check of a process model, the tool adopts the established practice of mapping a process model into a temporally-equivalent temporal constraint network such as the Simple Temporal Network with Uncertainty (STNU) or the Conditional STNU (CSTNU) (to represent processes with no conditional executions, resp. processes including conditional executions) and execute a constraint propagation procedure on it, which returns `true` if the network is dynamically controllable, `false` otherwise [1].

3 System Overview

TemporalBPMN-ID is an extension of the open source tool Camunda modeler (https://camunda.com) for modeling BPMN process diagrams. The extension for defining events, temporal parameters, durations, and temporal constraints is realized by annotating the BPMN source. The system has a client-server architecture: the client provides the user interface for defining time-aware process models; the server takes care of executing a DC-check for the model being defined, based on temporal constraint networks. The DC-check is performed whenever a significant change in the process model is detected. Significant changes are changes in the model that determine a change in the temporal aspect, i.e.:

- Adding or modifying a control flow element duration
- Removing a control flow element
- Adding or removing an event
- Adding, removing, or modifying a temporal constraint

The communication between client and server is REST-based. Whenever a significant change in the process model is detected, it is communicated from the client to the server. With the communication of significant changes, the server maintains an up-to-date temporal constraint network that encodes the temporal aspect of the process model by applying the mapping rules for mapping into an STNU with temporal parameters (PSTNU, [3]) or into a CSTNU (e.g., [6]).

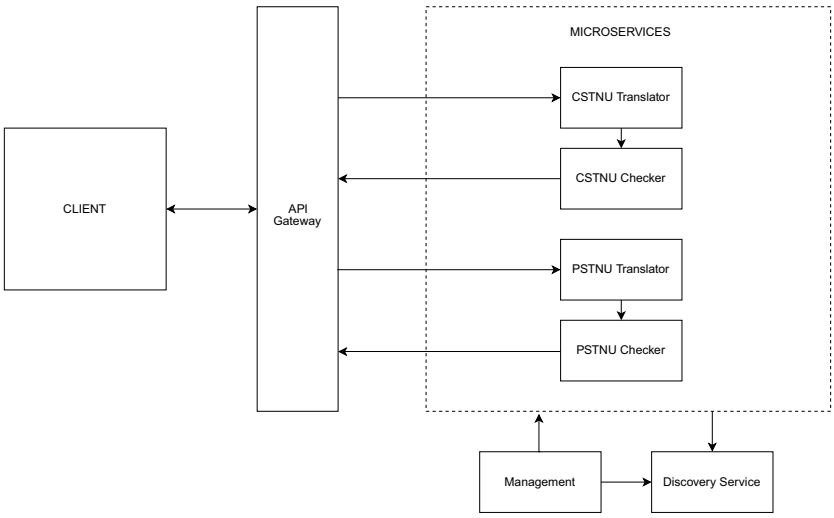

Fig. 1. Architectural diagram of the implemented system.

For the server, the implementation follows a microservice architecture, which allows for decoupling the different components and features. This allows plugging in different checking algorithms based on different data structures, e.g., PSTNU or CSTNU, which make use of different checking procedures, to be able to formally encode various temporal elements for temporal reasoning.

Fig 1 shows the architectural diagram for the implemented system.

4 Use Case

Here, we give an overview of a use case of modeling the following time-aware business process from the medical domain. The process starts with a blood sampling (taking 2 to 5 minutes). Then, in parallel, an MRI (Magnetic Resonance Imaging) scan is performed (20 to 45 minutes) and blood analyses are carried out (30 to 45 minutes). Finally, a diagnosis report is filled out (3 to 10 minutes). Usage of the MRI instrumentation must be notified at least 10 minutes in advance. The whole process, including inter-task delays, must take 90 minutes at most. All activities are contingent, and gateway executions take no time.

The steps taken by a process designer using our tool are the following:

1. First, the designer adds a pool modeling the clinic.
2. The designer adds the process start and end events to the pool. Each event introduction triggers a communication from the client to the server, requesting to create a new PSTNU node corresponding to the created event.
3. The designer adds various gateways and activities, and specifies the name, duration type, and duration range for the activities.

4. The designer connects the start event to the first activity with a control flow edge. Now that the two elements are connected, the tool detects that the time-aware process is in a new meaningful state and sends a new command to the server. This creates two new PSTNU nodes per activity (one for the start and one for the end of it), a contingent link between each pair of such nodes (representing the duration bounds), and a regular edge connecting the node for the process start event to the node for the first activity start event.
5. The designer adds an upper-bound constraint from the process start event to the process end event with bound 90 to model the requirement of the maximum allowed duration. A command is sent to the server to add a corresponding PSTNU edge between the nodes for the start and end of the process. Afterwards, the server starts the execution of a full constraint propagation procedure, which realizes a DC-check for the process. The procedure returns `true`, signaling that the process is DC, and such a result, along with the derived edges, is sent to the client, which displays *The process is DC*.
6. The designer adds the remaining control flow connectors, each time triggering a communication and a DC check like above. The process is still DC.
7. The designer adds an upper-bound constraint edge from the process start event to the start of the first activity. Now, due to the received derived edges after the full constraint propagation, the client knows that a derived edge exists between these two events. The corresponding value of 30 is shown automatically to the designer before she starts typing her intended constraint bound. The shown value is the maximum allowed value for such a constraint, i.e., any value larger than 30 violates the DC property of the process. It is now up to the designer to choose whether to keep such a value or to specify a different (smaller) one, in which case a new DC-check would be performed.
8. The designer adds a parameter node P to represent the time of notification of MRI instrumentation use since this is not a process activity. Then, she adds a lower-bound constraint from P to the start event of task M, with a bound of 10. This triggers the addition, to the PSTNU, of a new node and an edge connecting it to the node for the start of M. The server updates the PSTNU and executes a new full constraint propagation to compute potential new implicit constraints, returning `true`.
9. The designer adds an upper-bound constraint edge from process start to P. Again, the constraint is implicitly known from the propagation at the server, and the corresponding bound of 25 is presented to the designer, who accepts the value for the constraint. This value instructs process stakeholders on which is the maximum admissible value w.r.t. the process start for the MRI use notification, given all the other temporal requirements in the model.

Fig. 2 shows the view presented to the designer at the end of the above steps.

5 Conclusion

In this paper, we have introduced a novel tool for the interactive design of time-aware business processes. The tool extends the functionalities of the popular tool

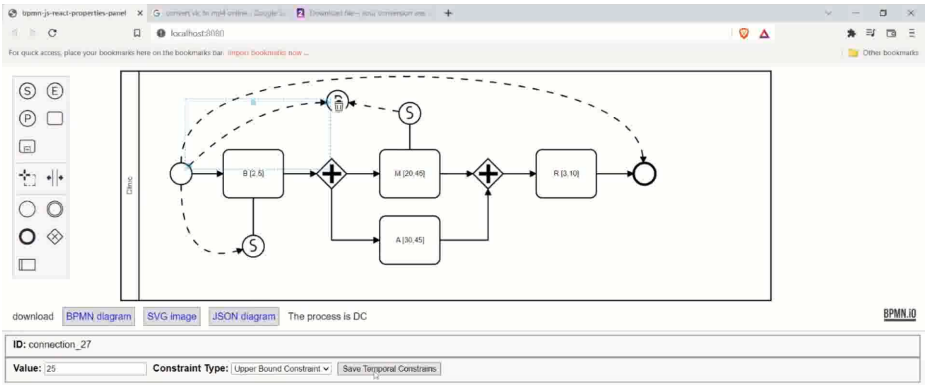

Fig. 2. Screenshot of the tool view after defining the process in the use case.

Camunda modeler with support for modeling the temporal aspect of processes. Additionally, the tool automatically performs an online DC-check on the process model during its design, providing the designer guidance on how to complete the temporal aspect definition. With the proposed tool, we provide process designers with a new means to define time-aware business process models that offer dynamic controllability guarantees. The microservice architecture enables further extensions with new algorithms to be used depending on the meta-model employed for time-aware processes and the temporal qualities to be checked.

References

1. Combi, C., Posenato, R.: Towards temporal controllabilities for workflow schemata. In: Temporal Representation and Reasoning (TIME), 2010 17th International Symposium on. pp. 129–136. IEEE (2010)
2. Combi, C., Pozzi, G.: Temporal conceptual modelling of workflows. In: Conceptual Modeling-ER 2003, pp. 59–76. Springer Berlin Heidelberg (2003)
3. Eder, J., Franceschetti, M., Köpke, J.: Controllability of business processes with temporal variables. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 40–47. ACM (2019)
4. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: Advanced information systems engineering. pp. 286–300. Springer (1999)
5. Franceschetti, M., Eder, J.: Determining temporal agreements in cross-organizational business processes. Information and Computation **281**, 104792 (2021)
6. Franceschetti, M., Eder, J.: Computing ranges for temporal parameters of composed web services. In: Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services. pp. 537–545 (2019)
7. Lanz, A., Reichert, M., Weber, B.: Process time patterns: A formal foundation. Information Systems **57**, 38–68 (2016)
8. Ocampo-Pineda, M., Posenato, R., Zerbato, F.: Timeawarebpmn-js: An editor and temporal verification tool for time-aware bpmn processes. SoftwareX **17**, 100939 (2022)