Temporal Variables for Time Modeling in Business Processes

Marco Franceschetti

Department of Informatics-Systems, Alpen-Adria-Universität Klagenfurt, Austria
firstname.lastname@aau.at http://isys.uni-klu.ac.at

Abstract. The modeling of temporal aspects in business processes is based on the representation of time as a mere property of control-flow constructs. Other timepoints cannot be represented, which limits the types of temporal constraints that modelers can express. Additionally, such representation of time hinders modularization and support for privacy in distributed processes. Here we propose a new process metamodel in which temporal aspects are explicitly modeled with temporal variables. We show that our metamodel gives modelers increased expressiveness to represent temporal constraints. We then give indications on how our metamodel can be used for overcoming the identified gaps.

Keywords: Business processes, Time modeling, Temporal variables

1 Introduction

Effective business processes management requires expressive models to capture all the relevant aspects of processes. The correct and comprehensive modeling of temporal aspects plays a crucial role in this perspective. Extensive research has been conducted over time and processes in the last decades, corroborating the relevance of temporal aspects in business process modeling. Despite the numerous approaches to conceptual modeling and verification of timed processes, such as [3, 6, 5, 9, 11, 14, 16], expressiveness limitations still affect the modeling of temporal information and temporal constraints in processes. In the current approaches, the temporal dimension of a process is regarded to as a mere property of its activities, not directly accessible. This limits the spectrum of possible temporal constraints that one can express. We aim at filling this gap by introducing a language with increased expressiveness for defining timed processes.

Here we propose a process metamodel that allows to represent temporal aspects as data elements. The reason behind it is that processes manipulate data, and temporal information handled at runtime is data too. This explicit representation of time allows modeling additional temporal information which are unrelated to process activities, defining a more extensive set of temporal constraints, while still allowing the verification of temporal properties of a process.

This work is supervised by Prof. Johann Eder from the Information and Communication Systems research group at the Alpen-Adria-Universität Klagenfurt, Austria.

The remainder of this paper is structured as follows: in Sect. 2 we present a series of motivating examples to show the gaps that our work aims at filling. Sect. 3 presents our research questions and methods. In Sect. 4 we present the state of the art. In Sect. 5 we define our process metamodel and briefly show how we can evaluate it. Sect. 6 gives an overview of the next steps of work, and we draw conclusions in Sect. 7.

2 Motivating Examples

Most business processes comprise temporal requirements to be fulfilled for a correct execution. Temporal properties of such processes can be checked at design time. However, there are real-world scenarios for which no mapping of temporal information or constraint is possible with the current modeling approaches. Here we show examples of such cases.

- *Temporal information unrelated to control flow elements.* Since existing approaches to time modeling only consider temporal aspects related to control flow elements, no other timepoints can be referred to. Suppose a medical scenario in which a patient comes to the hospital for urgent treatment. The treatment is possible only if the patient has not taken a specific medicine in the previous 24 hours. The medicine intake is not an activity of the treatment process, since it might have happened before. Therefore, the temporal information related to it is not part of the process. While in the real world it could be obtained from an interview with the patient, in a process model there would be no way to represent and directly refer to such a temporal information. More in general, with the existing approaches, all temporal information that is not bounded to an executable element is not representable and cannot be used for defining temporal constraints.
- *Temporal constraints across modules.* The advantages of modularizing processes span from the ability to reuse existing processes inside new ones, to the simplification of the overall process models thanks to abstraction. The presence of temporal constraints crossing modules, however, nullify these advantages. A temporal constraint involving a timepoint t_i within a module and one outside of it would require the outer process to have access to the module internals in order to refer to t_i, because it is a property of an activity in the module. Moreover, the outer process could require to dynamically constrain durations inside of the module, in order to satisfy some overall temporal constraint. In such a case there is no way to convey temporal information from the outside process to the internals of the module.
- *Temporal constraints across distributed processes.* A distributed process is realized by the interplay of local processes executed by different process partners; no central authority exists for computing a global schedule, and a shared-nothing architecture is assumed. The presence of temporal constraints across local processes then requires the access to the internals of those processes by other process partners, which is undesirable since it means a lack of privacy and a potential exposition of business secrets. Similarly to the

case of modules, currently no way to directly refer to temporal information originating outside of a local process is possible either.

3 Research Questions and Methods

The research questions that arise by looking at our examples, and that we try to answer with this work are the following:

1. *How can we explicitly model temporal aspects of business processes as data elements such as temporal variables?*
2. *What are the semantics of such process models?*
3. *Which expressive power will be enabled in (possibly modularized or distributed) process models, if temporal variables are introduced?*
4. *How can we define the correctness and other temporal properties, such as controllability, for (possibly modularized or distributed) process models, if temporal variables are introduced?*
5. *How can we check the correctness and other temporal properties, such as controllability, for (possibly modularized or distributed) process models, if temporal variables are introduced?*

The first contribution of this project is to provide a formalism with increased expressiveness for defining timed business processes, overcoming the expressiveness issues presented in Sect. 2 and allowing the modeling of a broader range of real-world processes. By applying design science methods, a language for defining (possibly modularized or distributed) process models with temporal variables will be produced, based on the one we presented in [10]. Temporal variables will be used to represent timepoints and durations; temporal constraints will be stated over them. Production of formal definitions, theorems and proofs will be the method applied for defining the metamodel semantics. The evaluation of the metamodel will be based on its implementation, and the design of real-world-based process models. Algorithm design will be the method for generating data interfaces for modularized and distributed processes in presence of temporal constraints; proofs of soundness and completeness will be given for validation, and implementation will be done to assess the viability of the approach.

An additional contribution is the production of a set of tools for assessing properties of processes expressed with the proposed language. Production of formal definitions, theorems and proofs will be the method applied for defining the notions of correctness and controllability of processes expressed with the proposed language. Algorithm design will be the method for checking such properties, also in modularized and distributed processes. Also in this case, proofs of soundness and completeness will be given for validation, and algorithm implementation will assess the feasibility of the approach.

4 State of the Art

Numerous previous works considered the incorporation of temporal aspects in workflows. The modeling of timepoints in BPMN is considered in [8], where

Time-BPMN is introduced to tackle the modeling of temporal constraints and dependencies. The work defines attribute and properties extensions and their depiction. Arbitrary timepoints, external to process activities, however, are out of the modeling scope. Arbitrary events are considered in [17], however, they are always associated to either state changes of objects or triggering of operations in the enactment environment. Thus, expressiveness is still limited. [2] surveys various approaches to the modeling of temporal constraints in business processes. None of those approaches explicitly models temporal aspects through temporal variables. The focus of [7] is on the build- and instantiation-time computation of activity deadlines to allow process execution without violating temporal constraints and the overall process deadline. The considered temporal constraints are upper- and lower-bound constraints. The adopted process metamodel, however, does not represent temporal aspects of activities in an explicit manner as we do here, and modularization and distributed processes are not considered. A formal framework for time modeling in production workflows is introduced in [14], along with algorithms for CPM-based calculation of minimum and maximum process durations and for constraints verification. The considered timepoints, however, only refer to process elements and there is a lack of support for additional timepoints. Attention to temporal constraints is given also in [3], where a temporal conceptual model for workflows is proposed, and three classes of temporal constraints are defined: task constraints, schedule-task constraints, and inter-task constraints. However, all considered timepoints for constraint definition relate to process tasks: no additional external event is contemplated. [12] focuses on the representation and support of modularized processes. The basis of the proposed representation for processes are Temporal Constraint Networks (TCNs, [4]), to which process models are abstracted for calculating temporal properties, such as controllability. Mapping processes to TCNs translates start and end events of activities to TCN nodes. However, such an approach ignores the existence of arbitrary timepoints. Temporal constraints on data are important also for temporal databases [15], however the focus is limited to their satisfiability, not addressing controllability aspects.

5 Process Metamodel

Since our aim is to allow expressing additional temporal aspects in process models, we need to address time at metamodel level. We propose a metamodel that aims at overcoming the expressiveness issues discussed before. It includes a minimal set of control-flow constructs that allow to model the most common control patterns [1]. We consider acyclic workflow nets composed of nodes connected with directed edges. Nodes can be activities, xor-splits, xor-joins. Xor-splits have exactly two outgoing edges, and xor-joins have exactly two incoming edges. The semantics is that in a single process instance only one successor of a xor-split is executed, as well as only one predecessor of a xor-join. All the other nodes have the implicit semantics of and-splits and and-joins, meaning that all edges leaving a node that is not a xor-split are followed for execution, and all predecessors of a node that is not a xor-join have to be executed before that node can execute. Activities can be either *contingent* or *noncontingent*. Contingent activities have fixed minimum and maximum duration, but their actual runtime duration cannot be controlled; instead, it can only be observed when the activity is executed. An example of contingent activity is a bank money transfer, for which some time between one and four days is required but there is no way to control it. Noncontingent activities have a fixed minimum and maximum duration as well, but the actual duration can be controlled by a human agent. Durations and other timepoints are explicitly represented as data elements which can be read or written by activities. Constraints can be stated to express duration restrictions between timepoints.

5.1 Process Graph

Since we consider time as data, following our previous work on handling data in distributed processes [10], we represent processes with process graphs.

Definition 1 (Process Graph). *A process graph P is a tuple (N, E, Γ, V, C), where:*

- *N is a set of nodes, each with its d_{min} and d_{max}, and sets r and w of read, resp. written variables;*
- *E is a set of edges, each $\in (N \times N)$;*
- *Γ is a set of propositional letters;*
- *V is a set of variables, partitioned in V_d (generic data variables) and V_t (temporal variables);*
- *C is a set of constraints, partitioned in UBC (upper-bound) and LBC (lower-bound). Constraints are in the form (s, d, δ), with $s, d \in V_t$, $\delta \in \mathbb{N} \cup V_t$.*

Nodes can be of different types: *start*, *end*, *activity*, *xor-split*, *xor-join*. Each node n has two associated temporal variables representing its minimum and maximum durations: $n.d_{min}$ and $n.d_{max}$. Furthermore, it has two temporal variables representing its start and end times at process runtime: $n.s$ and $n.e$, respectively. The following inequality must hold for all nodes n: $0 \leq n.d_{min} \leq n.e - n.s \leq n.d_{max}$. Each node n is also associated with two sets of variables $n.r, n.w \subseteq V$ containing the variables read and written in it.

Edges connect nodes and define the control flow of a process. We use edges to define the sets of predecessors, resp. successors of a node n as: $n.pred = \{m \in N \mid \exists(m, n) \in E\}$, $n.succ = \{m \in N \mid \exists(n, m) \in E\}$. A node n with $\|n.pred\| = 0$ is a *start node*; a node n with $\|n.succ\| = 0$ is an *end node*.

We use propositional letters to construct labels. Labels indicate through which paths a certain node can be reached from the start node. Different paths are possible due to the presence of decisions taken at xor-splits. For each xor-split node xs we associate a propositional letter $\lambda(xs) = \mu \in \Gamma$. The two outgoing edges e_1 and e_2 of a xor-split xs have labels $e_1.\gamma = \lambda(xs)$ and $e_2.\gamma = \neg\lambda(xs)$, respectively. All other edges e have label $e.\gamma = True$. Each node n of a process graph has a label $n.\Lambda$ which is defined depending on the type of the node:

- $n.\Lambda = \{True\}$ if n is a start node;
- $n.\Lambda = \bigcup_{m \in n.pred}(m.\Lambda)$ if $n.type =$ xor-join;
- $n.\Lambda = \otimes\{m.\Lambda \wedge (m,n).\gamma \mid (m,n) \in E\}$ for any other node, where the operator $\otimes$ is defined as the cross-conjunction of a set of sets Λ_i of conjunctive terms: $\otimes\Lambda = \{\lambda_1 \wedge \cdots \wedge \lambda_n \mid \lambda_i \in \Lambda_i, 1 \leq i \leq n\}$.

The set V is partitioned in the set of generic (data elements) variables V_d and the set of temporal variables V_t. Temporal variables encode temporal information such as durations, timepoints, deadlines. We define three partitions of V_t, depending on the type of information a temporal variable represents.

1. V_{tn} is the set of temporal variables derived from all nodes in the process graph: $V_{tn} = \{n.d_{min} \mid n \in N\} \cup \{n.d_{max} \mid n \in N\} \cup \{n.s \mid n \in N\} \cup \{n.e \mid n \in N\}$;
2. V_{td} is the set of temporal variables referring to temporal information that is written in some node in the process and is not derived as a temporal property of that node;
3. V_{te} is the set of temporal variables given by the environment, such as deadline for process termination (temporal variable Ω), maximum time of start for a node, elapsed time since process start (temporal variable η).

Each $v_t \in V_t$ is associated with a label $\phi(v_t)$, depending on its type:

- $\phi(v_t) = n.\Lambda$ if $v_t \in V_{tn}$;
- $\phi(v_t) = \bigcup_n n.\Lambda, v_t \in n.w$ if $v_t \in V_{td}$;
- $\phi(v_t) = True$ if $v_t \in V_{te}$.

The label determined by ϕ corresponds to the values of the decision variables under which a temporal variable is correctly initialized.

Constraints reflect restrictions on the values that can be assumed by variables at runtime. Here we concentrate on temporal constraints, as they are the ones that most influence the execution of a process and determine if certain temporal properties, such as controllability, hold. Two types of temporal constraints exist: *upper-* and *lower-bound*. Upper-bound constraints express maximum allowed durations between events, while lower-bound constraints express minimum required durations between events. The set C of constraints is therefore partitioned into the sets UBC and LBC of upper-, resp. lower-bound constraints. The semantics of a constraint $c = (s, d, \delta)$ is defined after the partition it belongs to:

- $d - s \leq \delta$ if $c \in UBC$;
- $d - s \geq \delta$ if $c \in LBC$.

5.2 Well-Formed Process

For space reasons, we informally define the requirements for a process to be well-formed, in order to avoid ill process definitions. In a nutshell: each xor-join node must not be reached by two different paths at the same time (disjoint paths); for all nodes that are not xor-joins their predecessors must be compatible; for each temporal constraint, there must be at least one path in which both the involved temporal variables are correctly initialized; each temporal variable must not be written more than once in a same process instance.

5.3 Metamodel Evaluation

We sketch the steps needed to evaluate our metamodel: we first need to show that it is able to express all the established time patterns [13] (with the exception of those that relate to loops, since loops are not contemplated in our metamodel). We can show that each time pattern can be constructed with our metamodel, proving that it is not less expressive than existing metamodels. Additionally, we can show that our metamodel brings additional expressiveness since it captures the situations pointed out in Sect.2 which existing approaches fail to model.

6 Next Steps

Having our process metamodel at disposal, we identify three main avenues for further work, guided by our research questions: the automated check for temporal properties, in particular controllability, the support for modularized processes, and the support for distributed processes.

6.1 Correctness and Controllability

Given a process model, it is interesting for modelers to determine its correctness and its controllability. For correctness here we refer to an assignment of timestamps to temporal variables which is not contradictory. The concept of controllability covers the problem of determining the possibility of executing a process without violating its temporal constraints. Temporal Constraint Networks, and their specialization which includes decision points and uncertain durations given by contingency (CSTNUs), are a widely used graph-based structure for representing timepoints and their dependencies. Since established methods exist for determining their controllability, we plan to develop a mapping to transform our process models into CSTNUs for controllability check. The idea is to map each temporal variable to a timepoint of a CSTNU, and each temporal constraint to an edge of the CSTNU. This method will allow to exploit the established results for controllability check in CSTNUs to determine the controllability of our process models. A formalization of the concepts of correctness and controllability through formal definitions will be provided. Algorithms for determining if a process is correct and controllable, as well as for mapping process models to CSTNUs, will be developed and implemented.

6.2 Modularized Processes

The explicit representation of temporal information opens new possibilities for modularized processes by allowing the expression of temporal constraints without exposing module internals. Temporal variables can be used both as input and output parameters of a module, and be referenced in constraints within the module or in the outer process containing it. With constraints, input parameters coming from the outer process can influence timepoints within a module, and output parameters from a module can influence timepoints in the outer process. The influence of parameters opens questions such as whether and under which conditions it is necessary to recompute temporal properties of modules for checking the ones of the outer process. Methods for determining the required data exchanges across modules, and the need to recompute temporal properties of modules when required, will be identified, and corresponding algorithms will be designed and implemented. Additionally, as it is possible to assign ranges for the timestamps referenced in temporal constraints crossing modules, algorithms to predetermine such ranges will be designed and implemented.

6.3 Distributed Processes

Our metamodel can be applied to distributed processes, allowing to keep local process internals hidden even in the presence of temporal constraints across local processes. In a distributed process, several participants enact their local processes in coordination with each other. The coordination between different local processes can be realized by introducing communication primitives to exchange data at required points, as shown in [10]. The communication primitives are activity nodes in which data is sent or received. A distributed process can therefore be represented through our process graph, by assigning each node to the corresponding process partner for execution. Temporal constraints across local processes can be realized by exchanging temporal variables using communication primitives. Means to automatically identify the required data exchanges will be investigated, and corresponding algorithms will be designed and implemented. Additionally, algorithms for the automatic generation of temporal ranges for executing local activities which would fulfil the global temporal constraints will be designed and implemented. The same will be done for algorithms aimed at verifying the overall temporal properties of distributed processes.

7 Conclusion

Despite temporal aspects being of humongous importance in business processes, most existing approaches neglect those which are not directly related to process activities, and provide a limited expressiveness for representing time. In this paper we have introduced a new metamodel for processes which is the first using temporal variables to represent temporal aspects. We have sketched proof that our proposed metamodel is able to capture all the compatible established time patterns, and that our explicit representation of time provides additional expressiveness by enabling the modeling of new temporal aspects and constraints. Future work includes the verification of temporal properties of processes, possibly by mapping our metamodel to CSTNUs, the automatic generation of interfaces for process modules and for distributed processes.

References

1. van der Aalst, W.M., Ter Hofstede, A.H., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and parallel databases **14**(1), 5–51 (2003)
2. Cheikhrouhou, S., Kallel, S., Guermouche, N., Jmaiel, M.: The temporal perspective in business process modeling: a survey and research challenges. Service Oriented Computing and Applications **9**(1), 75–85 (2015)
3. Combi, C., Pozzi, G.: Temporal conceptual modelling of workflows. In: Conceptual Modeling-ER 2003, pp. 59–76. Springer Berlin Heidelberg (2003)
4. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artificial intelligence **49**(1-3), 61–95 (1991)
5. Eder, J., Pichler, H., Vielgut, S.: An architecture for proactive timed web service compositions. In: Business Process Management Workshops. LNCS, vol. 4103, pp. 323–335. Springer (2006)
6. Eder, J., Gruber, W., Panagos, E.: Temporal modeling of workflows with conditional execution paths. In: Database and Expert Systems Applications. pp. 243–253. Springer (2000)
7. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: Advanced information systems engineering. pp. 286–300. Springer (1999)
8. Gagne, D., Trudel, A.: Time-bpmn. In: Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on. pp. 361–367. IEEE (2009)
9. Guermouche, N., Godart, C.: Timed model checking based approach for web services analysis. In: ICWS 2009. IEEE International Conference on Web Services, 2009. pp. 213–221. IEEE (2009)
10. Köpke, J., Franceschetti, M., Eder, J.: Analyzing data-flow implementations for distributed execution of inter-organizational processes. In: Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services. pp. 502–510. ACM (2017)
11. Lanz, A., Posenato, R., Combi, C., Reichert, M.: Controllability of time-aware processes at run time. In: On the Move to Meaningful Internet Systems: OTM 2013 Conferences. pp. 39–56. Springer (2013)
12. Lanz, A., Posenato, R., Combi, C., Reichert, M.: Controlling time-awareness in modularized processes. In: Enterprise, Business-Process and Information Systems Modeling, pp. 157–172. Springer (2016)
13. Lanz, A., Weber, B., Reichert, M.: Time patterns for process-aware information systems. Requir. Eng. **19**(2), 113–141 (2014). https://doi.org/10.1007/s00766-012-0162-3, http://dx.doi.org/10.1007/s00766-012-0162-3
14. Marjanovic, O., Orlowska, M.: On modeling and verification of temporal constraints in production workflows. Knowledge and Information Systems **1**(2), 157–192 (1999)
15. Ozsoyoglu, G., Snodgrass, R.T.: Temporal and real-time databases: A survey. IEEE Transactions on Knowledge and Data Engineering **7**(4), 513–532 (1995)
16. Pichler, H., Eder, J., Ciglic, M.: Modelling processes with time-dependent control structures. In: International Conference on Conceptual Modeling. pp. 50–58. Springer (2017)
17. Rolland, C.: A methodology for information system design. In: Proceedings of the May 4-7, 1981, national computer conference. pp. 583–589. ACM (1981)