SoftwareX 17 (2022) 100939

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

TimeAwareBPMN-js: An editor and temporal verification tool for Time-Aware BPMN processes

Mario Ocampo-Pineda [a], Roberto Posenato [a,*], Francesca Zerbato [b]

[a] Department of Computer Science, University of Verona, Italy
[b] University of St. Gallen, Switzerland

ARTICLE INFO

Article history:
Received 25 September 2021
Received in revised form 3 December 2021
Accepted 3 December 2021

Keywords:
BPMN
Temporal process modeling
Temporal constraint networks
Dynamic controllability

ABSTRACT

This paper presents TimeAwareBPMN-js, a graphical web-based editor for time-aware BPMN (Business Process Model and Notation) models that allows (1) creating and editing of BPMN processes enriched with temporal constraints, such as *contingent durations* and *conditions*, and (2) verifying that such constraints are well-defined and satisfy some (temporal) properties. The verification of temporal constraints is realized by plug-ins that can be easily added by the user thanks to the modular architecture of the application. Different plug-ins may verify different temporal properties. As a proof-of-concept, TimeAwareBPMN-js contains the *CSTNU plug-in* which verifies the *dynamic controllability* property, i.e., it checks, at design-time, whether there exists a run-time schedule for the process that satisfies all temporal constraints no matter how contingent durations and conditions are revealed during execution.

Code metadata

Current code version	v1.3
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00180
Code Ocean compute capsule	N.A.
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	JavaScript, Java
Compilation requirements, operating environments & dependencies	JVM $\geq$ 11, Node.js, bpmn-js-properties-panel, camunda-bpmn-moddle, diagram-js, inherits, jquery, min-dash, tiny-svg, xmlbuilder
Developer documentation/manual	https://gitlab.com/univr.di/TimeAwareBPMN/-/wikis/home
Support email for questions	mario.ocampo@univr.it

1. Motivation and significance

Business process models are widely used by organizations to design, analyze, implement, and control their business processes. Process models are typically specified with the help of process modeling languages, such as the Business Process Model and Notation (BPMN) standard [1]. Depending on the process modeling language used, process models can capture different perspectives. Among them, we find the *control flow*, i.e., the activities executed by a process and their order, the *data flow perspective*, and the *temporal perspective* [2]. This last one refers to the wide range of temporal properties and constraints that the process needs to satisfy during its execution.

The modeling and *verification of temporal constraints* in business processes have been a long-researched topic within the Business Process Management (BPM) community (see [3] for an updated survey). For example, there are approaches that combine extensions of BPMN [4,5] with timed automata (e.g., [6–8]) or Petri Nets (e.g., [5,9,10]), and approaches based on temporal constraint networks (e.g., [11–15]), or temporal logic (e.g., [16]). In [17], the authors elicited a set of ten time patterns covering temporal aspects relevant for the modeling and control of business processes and activities.

Among such patterns, *durations* and *time lags* between activities seem to be those occurring most often in practice. In

* Corresponding author.
E-mail addresses: mario.ocampo@univr.it (Mario Ocampo-Pineda), roberto.posenato@univr.it (Roberto Posenato), francesca.zerbato@unisg.ch (Francesca Zerbato).

https://doi.org/10.1016/j.softx.2021.100939

particular, durations allow constraining the time span needed for executing process elements [17]. In recent years, researchers have remarked on the need to discern between two kinds of duration constraints [15]: *controllable* and *contingent*. A controllable duration is relative to process activities (henceforth called *controllable activities*) that are managed by the agent executing the process. In contrast, a contingent duration is relative to other activities (henceforth called *contingent activities*) that are managed by an external agent and for which the agent executing the process can observe the duration only after the activity has been executed [11]. Usually, duration constraints are specified at design time as temporal intervals that define the minimum and maximum value of the duration period. Time lags, also called *relative constraints* [11], limit the time distance between the starting/ending instants of any two process activities (cf. TP1 in [17]) or arbitrary events (cf. TP3 in [17]).

Given a time-aware process model, i.e., a process model with temporal constraints specified, different properties can be checked at design time to ensure that the process is executed correctly. An interesting property is the *dynamic controllability*, which guarantees that it is possible to schedule all controllable activities in such a way that all the temporal constraints are satisfied given all the possible durations of the contingent activities.

Temporal constraint networks are among the most refined models for modeling and solving temporal problems [18,19], including dynamic controllability checking. A temporal constraint network is dynamically controllable if it is possible to determine a schedule that, given the activities executed so far, incrementally determines which activities can be started (and for the controllable ones, their duration) to satisfy all the temporal constraints.

In this paper, we present TimeAwareBPMN-js, a tool that enables the modeling and verification of time-aware BPMN models [20]. The modeling is realized by a GUI editor that allows modeling well-defined time-aware BPMN processes. The verification of the *temporal properties* is based on plug-ins that can check different temporal properties using techniques defined by the user. As a proof-of-concept, the tool features the *CSTNU plug-in*, which is able to verify the dynamic controllability of a time-aware BPMN process, converting the process into an equivalent Conditional Simple Temporal Network with Uncertainty (CSTNU) and exploiting the checking algorithm for it [21].

1.1. Related work

In the literature, there are many proposals for managing temporal aspects in BPMN processes. However, only a handful of them offer a tool that can be used directly to check the consistency/controllability of temporal properties. Below, we present an overview of such proposals in chronological order.

In [22,23], the authors proposed a transformation from an extended version of BPMN 2.0 to the Timed Communicating Sequential Process model (CSP+T). The proposed BPMN extension admits the definition of duration constraints for tasks and the relative constraints among activities. The main contribution of such an approach is the definition of formal semantics for timed BPMN elements in terms of CSP terms. Both proposals did not offer a ready-to-use tool for the transformation to CSP and its check.

In [7,20], the authors proposed to represent absolute and relative temporal constraints and to analyze the behavior of such extended BPMN models transforming them to timed automata. For the modeling, they propose to extend the Activiti Eclipse Designer (http://www.activiti.org/), while for the verification, they present only a method to support the transformation to timed automata without offering a tool. Once a timed automaton is determined, it can be analyzed by the UPPAAL Tiga tool [24] to check possible deadlocks or deadlines violations.

In [25], the authors proposed a tool, ATAPIS, that allows the management of business processes with many temporal properties/constraints described in [12]. The tool represents business processes in an extended ADEPT2 notation [26], where the specification of temporal properties is straightforward and intuitive. In ATAPIS, the dynamic controllability property of a business process is checked by transforming the model to a CSTNU instance [21] and using the checking algorithm for CSTNUs. Using ATAPIS it is possible to design, check, and manage possible inconsistencies in an integrated way without requiring any external tool or temporal-constraint-network knowledge.

In [27], the authors proposed an encoding of a subset of BPMN elements into Maude, a rewriting logic system, where they also propose how to represent duration constraints for the encoded tasks and flows. Such temporal constraints are defined as stochastic expressions, while probabilities are used to represent various forms of branching behavior in gateways. In the paper, the authors also presented some case-studies about checking timed processes but without providing a ready-to-use tool for designing and checking the models.

In [28], the authors proposed a formal semantics for the time-related elements of BPMN supporting different combinations of events, time information categories (date-times, durations, cycles), and the corresponding ISO-8601 descriptions as prescribed by the BPMN standard. Then, the proposed semantics is directly defined in terms of First-Order Logic and, then, translated in Alloy [29] to support the formal verification of processes. As the same authors stated, the proposed tool allows the semantics validation on only a subset of study-case models, but *"real-life models are often out of reach of Alloy Analyzer as the number of required states for an execution exceeds its capacity"*.

In Table 1, we provide a comparison of the above approaches for managing temporal aspects. For each approach, we report if it supports the modeling of BPMN processes, some commonly used time patterns [12], the checking of the dynamic controllability property, and, most importantly, whether they offer an integrated ready-to-use tool. Such a tool should allow users to design a process model with temporal constraints and verify temporal properties without requiring the users to make other external actions like transforming the model in other formats and/or using external applications. To our knowledge, only ATAPIS [25] and TimeAwareBPMN-js offer such a tool, with ATAPIS being based on ADEPT2 and TimeAwareBPMN-js based on BPMN.

2. Software description

TimeAwareBPMN-js is a web application that extends the bpmn-js toolkit—a BPMN editor developed by Camunda [31]—to model and verify time-aware BPMN models.

2.1. Software architecture

TimeAwareBPMN-js is a web client–server application where the client, a web application, is an advanced editor that communicates with a server where there are one or more temporal verification tools (developed as desktop applications that run on the server-side). In this way, it is possible to exploit desktop applications/libraries written in different languages that, otherwise, they cannot be executed inside a browser (client side).

In particular, the client is the BPMN web editor bpmn-js [31] extended (1) to model time-aware BPMN processes, (2) to offer different verification tools realized as plug-ins, and (3) to interact with a server application via `HTTP` protocol.

The server is a Node.js application that, at the first interaction, returns the BPMN editor and, then, handles the requests of the

Table 1
A summary of different approaches in the literature that support the modeling and verification of temporal constraints in business processes. The label "planned" means that the feature is currently under development.

Reference		Capel, Mendoza [23], Morales et al. [22]	Cheikhrouhou et al. [20,30]	ATAPIS [25]	Durán et al. [27]	Houhou et al. [28]	Our tool
Year		2011	2013	2016	2018	2021	2021
Formalism for temporal reasoning		CSP+T	Timed Automata	CSTNU	Maude	FOL	CSTNU & o.
BPMN support?		✓	✓	×	✓	✓	✓
Patterns	time lags	✓	✓	✓	✓	✓	✓
	activity duration	✓	✓	✓	✓	✓	✓
	fixed date element	×	×	✓	×	✓	planned
	time-dependent variability	×	×	✓	×	✓	×
	cyclic elements	×	×	✓	×	✓	planned
Dynamic controllability?		✓	×	✓	×	×	✓
Integrated tool?		×	partially	✓	×	partially	✓

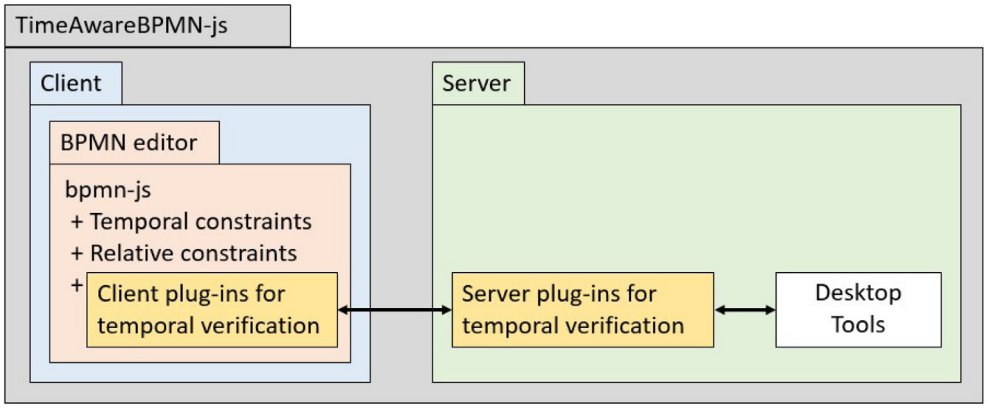

Fig. 1. Block diagram of the interaction between client and server plug-ins.

client by running server-side plug-ins for performing the verification of the temporal constraints according to the requested temporal property.

Fig. 1 displays the part of the block diagram of the TimeAwareBPMN-js related to the interaction between a client-side plug-in and the corresponding server-side plug-in to emphasize how the plug-ins work.

The interface of the application is composed of two sections:

1. A graphical editor that enables the modeling/editing of processes enriched with temporal constraints as detailed below;
2. A toolbar where it is possible to select which plug-in to activate and, according to the chosen plug-in, select the available actions.

Fig. 2 depicts a screenshot of the interface of TimeAwareBPMN-js, showing the graphical editor (region 1) and the toolbar containing buttons for the possible actions (region 2).

The palette a contains the BPMN elements that can be used to design a model. The ranges framed in blue, such as b, represent contingent duration constraints for BPMN elements like Tasks and catching Events. When the range has an error, e.g., the minimum duration is greater than the maximum duration, the frame turns red (e.g., d). The green-framed ranges c represent the duration constraints for elements like Gateways. Green color is to underline that such framed ranges can be restricted by the temporal verification algorithms to guarantee some properties, if it is necessary. On the contrary, blue framed ranges, i.e., contingent ones, cannot be modified, and the verification algorithms must guarantee that any values of such ranges can be used at run-time.

The edge icon e is a new button for adding an edge representing a relative constraint between any two BPMN elements [13]. The long-dashed-double-short-dashed edge f is an example of relative constraint between tasks `T1` and `T4`. The panel g shows the new tab of the element-attribute panel for editing temporal constraints of the considered element (e.g., in the figure, the tab shows the minimum duration and the maximum duration of the user task `T4`). The drop-down list h `Temporal constraint verification` allows selecting which verification plug-in to use. The possible actions (buttons on the line) are automatically updated based on the selected plug-in. In Fig. 2, the selected plug-in is CSTNU, featuring actions `Propagate conditions`, `Temporal verification`, `Download CSTNU`, and `Reset colors`. The `Browse` button i allows loading a time-aware BPMN model from a file, while the `Save` button j allows downloading the represented model as an XML document.

2.2. Software functionalities

The main and novel functionalities of TimeAwareBPMN-js are:

- The modeling of a relevant subset of BPMN elements, each of them augmented by temporal constraints or attributes useful to represent temporal constraints;
- The modeling of relative constraints between any pair of BPMN elements to represent different kinds of time lags [13, 17];
- The verification of temporal constraints and properties defined on a BPMN process by means of external tools integrated as plug-ins in the application.

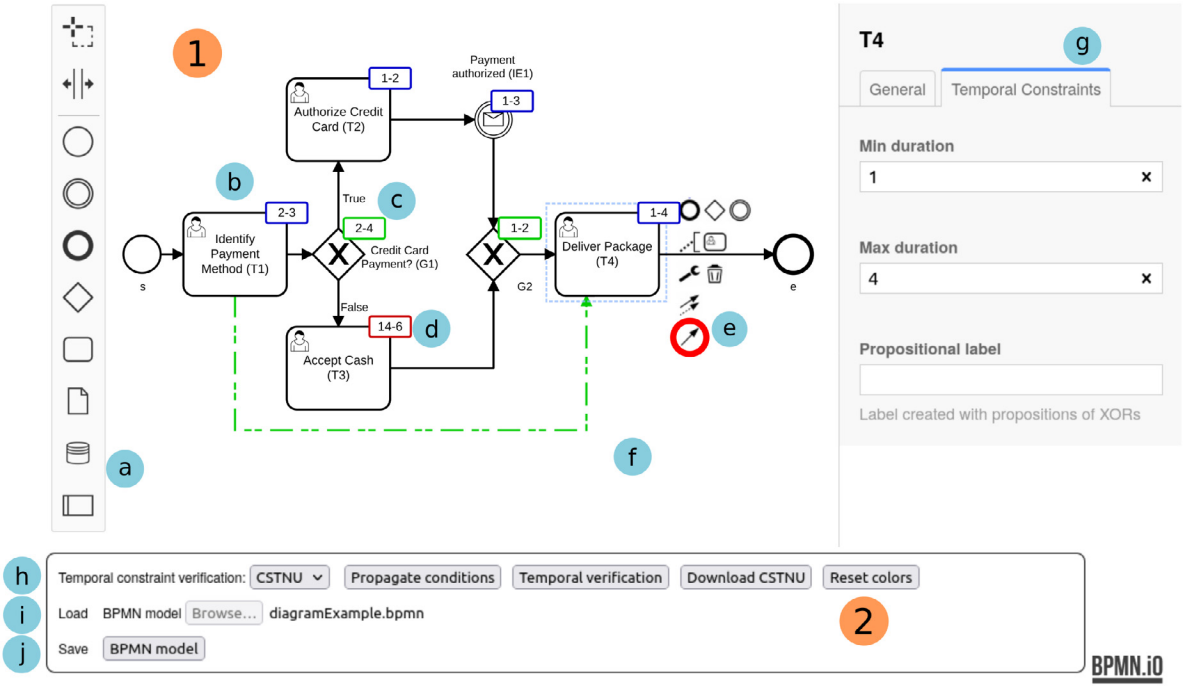

Fig. 2. Screenshot of the interface of TimeAwareBPMN-js.

The TimeAwareBPMN-js editor allows the management of a subset of basic BPMN elements enhanced with temporal constraints and attributes. In line with literature [6,13], we consider enhancing with temporal constraints elements that allow modeling basic control flow patterns and focus on *flow objects* commonly used in real-world scenarios. Besides, we allow the user to model basic data, swim lanes, and artifacts elements, although we do not enhance them with temporal constraints.

Table 2 summarizes the complete set of BPMN supported by our tool, reporting, for each BPMN element, which temporal constraints and attributes can be defined on it. Columns *Min duration* and *Max duration* represent the two constraints that limit the time range for executing the element. Column *Contingent?* specifies if the duration constraint is assumed to be contingent or not. *A propositional label* is a string of boolean literals (e.g., p, $\neg p$) that represents the condition for applying the temporal constraints of the element. For example, the propositional label $p\neg q$ says that the temporal constraints relative to the element must be observed when condition p is true and condition q is false. An empty propositional label indicates that the temporal constraints of the element must always be observed when the element is executed. The possible boolean conditions of a model are defined in exclusive (XOR) gateways and are set at run-time when the gateway is executed. Our model of time-aware BPMN processes [13] admits only binary XOR gateway for representing alternative flows of execution[1]. At run-time, when a XOR gateway is executed, the system evaluates the *condition* associated to the XOR gateway and, then, executes the branch associated to the truth value of the condition. The truth value of the condition observed in a XOR gateway is stored in a boolean variable, whose name can either be chosen by the user in the `Temporal Constraints` tab of the `Properties` panel of the exclusive gateway or assigned directly by the program. The name of such a variable is stored in the property *Observed proposition* (see Table 2). For example, in Fig. 2, different tasks must be executed based on which payment method is chosen. From a control flow perspective, the different tasks must be put on two alternative flows. The XOR Split gateway observes the condition "Credit Card Payment?" and stores the truth value in the variable p. In this way, the temporal constraints about task "Authorized Credit Card" must have a propositional label equal to p for being considered only when that execution flow is taken. In general, implicit propositional labels are determined automatically by the program for all elements. Such implicit propositional labels can be customized by the user to expand or restrict the applicability of the associated constraints.

In this version of the TimeAwareBPMN-js editor, time is assumed to be discrete, and temporal constraints values are represented as an integer without an explicit time granularity. Therefore, the user must fix a proper time granularity (e.g., minutes, seconds) for the elements that allows representing all temporal constraints as integer values.

2.2.1. Relative constraints

The TimeAwareBPMN-js editor allows specifying relative constraints between pairs of flow objects (i.e., activities, events, and gateways), as described in [13]. A relative constraint limits the temporal distance between two given instants of an execution. It is specified giving a temporal interval, the two relative instants, and a propositional label (for possibly customizing its applicability). Given a pair of flow objects, the first instant can be the *start* or the *end* of the first element, while the second instant can be the *start* or the *end* of the second element. Only for the BPMN *Start (none)* and *End (none)* elements, the start and the end instants are the same because such flow elements are defined as instantaneous events. For example, in Fig. 2, the edge (f) between tasks `T1` and `T4` is a relative constraint that was drawn using the button (e). The details of the constraint are shown in its `Temporal Constraint` tab (depicted in Fig. 3): it limits the temporal distance between the start instant of `T1` and the end instant of `T4` to be a value in the interval [2, 18].

[1] If there are more than two alternative flows of execution, it is possible to represent them putting two or more binary XOR gateways in cascade.

Table 2
Complete list of BPMN *Process* elements supported in the tool, including *flow objects* enhanced with temporal constraints. ✓: the constraint can be configured; ×: the constraint is not applicable.

BPMN element	Min duration	Max duration	Propositional label	Contingent?	Observed proposition	Source/target for a relative constraint?
Tasks						
User Task	✓	✓	✓	yes	×	✓
Service Task	✓	✓	✓	yes	×	✓
Script Task	✓	✓	✓	no	×	✓
Send Task	✓	✓	✓	yes	×	✓
Receive Task	✓	✓	✓	yes	×	✓
Events						
Start (none)	×	×	×	no	×	✓
End (none)	×	×	×	no	×	✓
Intermediate Catching						
Timer	✓	✓	✓	no	×	✓
Message	✓	✓	✓	yes	×	✓
Signal	✓	✓	✓	yes	×	✓
Intermediate Throwing						
Message	✓	✓	✓	no	×	✓
Signal	✓	✓	✓	no	×	✓
Gateways						
Parallel (AND)	✓	✓	✓	no	×	✓
Exclusive (XOR)	✓	✓	✓	no	✓	✓
Connecting Objects						
Sequence Flow	✓	✓	×	no	×	×
Association	×	×	×	no	×	×
Swimlanes						
Pool (single)	×	×	×	no	×	×
Lane	×	×	×	no	×	×
Data						
Data Object	×	×	×	no	×	×
Data Store	×	×	×	no	×	×
Artifacts						
Text annotation	×	×	×	no	×	×

2.2.2. Verification of temporal properties

Verifying temporal constraints in a time-aware BPMN process means checking that the definitions of such constraints are coherent at minimum. For example, if two activities must last at least 5 min each (specified by two temporal constraints), it is not possible to have a constraint that limits the overall execution time of both activities to 9 min.

However, this makes the verification task more challenging since the temporal constraints can have different properties (controllable, contingent, conditional). As already introduced in Section 1, the verification task could be to verify whether the model is *dynamically controllable*, i.e., it is possible to schedule all controllable tasks so that all temporal constraints are satisfied for whatever duration of the contingent tasks.

Another verification task could be to verify which tasks can be completed without any constraint violation within a given deadline.

In other words, such a temporal verification task should be flexible for checking different temporal properties of the model.

Therefore, we chose to delegate the execution of the verification task to plug-ins, i.e., software modules that can be added to TimeAwareBPMN-js to verify whether a temporal property is satisfied by the current BPMN model. Each plug-in can run an external software/library for realizing the check thanks to an API provided by TimeAwareBPMN-js.

As a proof-of-concept, TimeAwareBPMN-js is distributed with a plug-in, *CSTNU plug-in*, that can check the dynamic controllability of the model. In detail, the CSTNU plug-in transforms the current BPMN model into a corresponding instance of the CSTNU model following the method presented in [13]. Then, it uses an external library, CSTNU Tool [32], for checking the dynamic controllability of such a network.

In [13], the authors considered a richer time-aware BPMN model where a choice can be either an *observation*, if its value is decided by the environment, or a *decision*, if its value is decided by the executor of the process. Then, they proved that the dynamic controllability of a process of such a time-aware BPMN model is equivalent to the dynamic controllability of the corresponding CSTNUD instance. A CSTNUD instance is a CSTNU one where there are also boolean variables representing *decisions* that must be managed differently compared to *observations* [33].

Since in TimeAwareBPMN-js we do not represent *decisions*, the proposed instance conversion always determines a CSTNUD with zero decisions, i.e., a CSTNU. Therefore, the dynamic controllability for our time-aware BPMN models is equivalent to the dynamic controllability of the corresponding CSTNU instances.

The dynamic controllability check in the CSTNU plug-in also minimizes non-contingent duration ranges, removing all duration values that cannot be used. Then, if the model is dynamically controllable, the plug-in updates all temporal constraints (but the contingent ones because they cannot be modified) and shows the result to the user. If the model is not dynamically controllable, the plug-in shows a message to the user providing details about the inconsistencies found in the process model.

In general, the CSTNU plug-in verification-time is around few minutes for BPMN models having up to 20 activities. The time depends heavily on the number and on the disposition of XOR gateways. Indeed, in [21], the authors showed that there exist

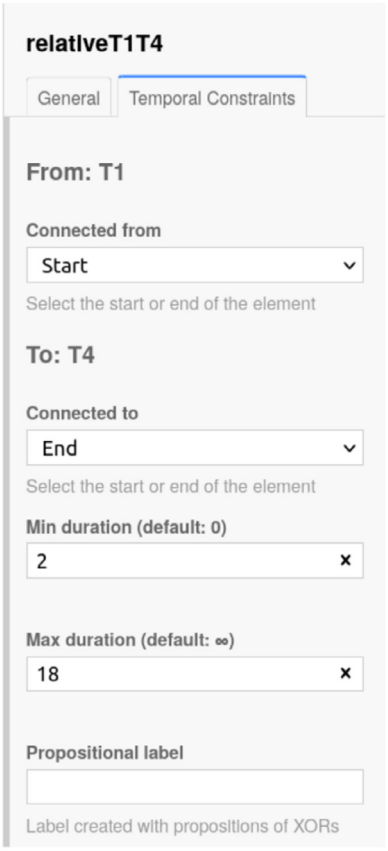

Fig. 3. Temporal Constraint tab of the relative constraint between T1 and T4.

some random instances having 10 activities and up to 5 XOR gateways that require up to 10^3 seconds to be verified (worst-case analysis). Such instances are characterized by the fact that the XOR gateways are in sequence and that there are global relative constraints that depend on all the XOR choices.

2.2.3. Implementation details

In this section, we give some implementation details about how the plug-ins work. Let us assume that a user wants to use an external library offering some methods that are useful for checking some temporal properties in time-aware BPMN processes. Such a library can be used in TimeAwareBPMN-js by adding two plug-ins, one for the client and one for the server.

The client-side plug-in goals are to: (1) provide buttons associated to specific actions through which a user can request the verification of some temporal properties, (2) transmit the request with all necessary data to the server, and (3) receive and show the results.

The server-side plug-in goals are to: (1) accept the verification requests, (2) realize the verification calling the methods of the associated library, and (3) return the results.

To easily integrate such pairs of plug-ins, TimeAwareBPMN-js offers a JavaScript API for allowing a plug-in to interact with the application. It is sufficient to implement the two plug-ins as JavaScript modules using the TimeAwareBPMN-js API and save them in specific directories for having the plug-in available immediately.

3. Illustrative examples

In this section, we give an example of how the tool can be used for verifying a time-aware BPMN model. The general flow of actions can be organized in the following steps.

1. The user can create a new BPMN model by adding the desired elements and connectors from the palette and setting the temporal constraints using the temporal-constraints panel provided in the web editor. Alternatively, the user can load a BPMN model using the *Load* button in the toolbar.
   For example, in Fig. 2 the user created a simple process model with a start event (`s`), four user tasks (`T1`, `T2`, `T3`, and `T4`), two exclusive gateways (`G1` and `G2`), a message event (`IE1`), and an end event (`e`). Minimum and maximum duration constraints have been specified for all tasks, intermediate events, and gateways using their `Temporal Constraints` panel, (in Fig. 2 there is the example of `T4` panel). Moreover, the process contains also a relative constraint between task `T1` and task `T4`.
2. Once all the temporal constraints have been specified, the user can select the plug-in to verify the temporal constraints of the model using the `Temporal constraint verification` drop-down list in the toolbar. In Fig. 2, the user selected the CSTNU plug-in.
3. Then, the user can select one of the actions available in the plug-in. For example, `Temporal verification` checks whether all temporal constraints are well-defined. If a temporal constraint is not well-defined, e.g., `min duration > max duration`, the tool reports an error showing a window describing the error and the constraints involved. An example is shown in sub-panel **a** of Fig. 4.
   If all constraints are well-defined, the tool verifies the dynamic controllability of the model converting the process into an equivalent CSTNU instance and verifying its dynamic controllability.
4. After running the temporal verification, the result is presented as a message, and the process model is updated. Sub-panels **b-d** in Fig. 4 depict three possible outcomes of the verification: the process is dynamically controllable, and there is no need to update it (**b**), the process is dynamically controllable, and some constraints need to be updated (**c**), and the process is not dynamically controllable (**d**). If the model is updated, then the frame color of the updated element changes. If the model is not dynamically controllable, the tool updates the frame color of the elements that determine the "non-controllable" status.

4. Impact and conclusions

In this work, we described TimeAwareBPMN-js, a web application for modeling and verifying time-aware BPMN models.

As regards the impact of the application, we envision it can serve different purposes and support researchers in various application settings:

- **Introduction to time-aware BPMN models.** The application can help researchers in the study of how temporal constraints can be modeled within BPMN models. Indeed, a researcher can load any BPMNv2 XML file representing a process model and enrich it with different temporal constraints to represent run-time requirements. Since the application allows verifying temporal constraints in a simple way, such study results are to be facilitated. Moreover, the application allows one to save all temporal constraints as extension elements in the BPMN file to simplify their distribution.
  Indeed, the study of temporal constraints within a BPMN model is not easy as it could seem. For example, the interplay of relative constraints and contingent ones having different propositional labels can easily lead to run-time

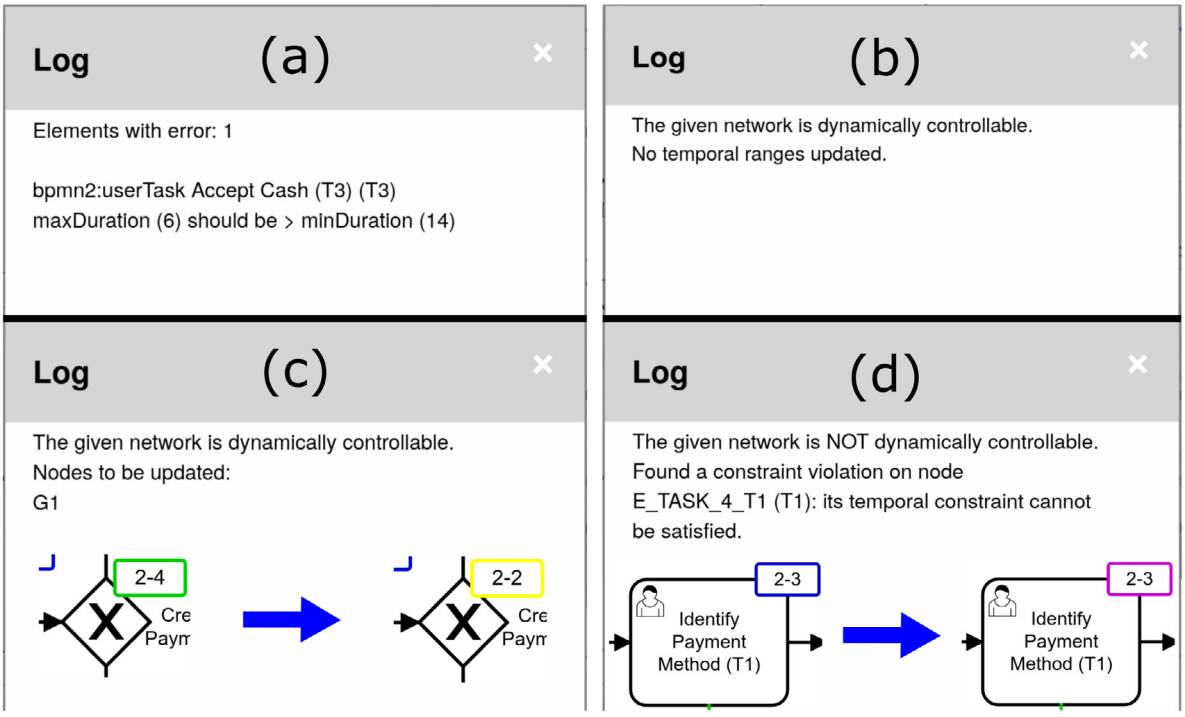

Fig. 4. Example of messages displayed by `Temporal verification` action. **(a)** Errors in the definition of temporal constraints. **(b)** Successful `Temporal verification` result without the need to modify the model. **(c)** Successful `Temporal verification` result with updated elements, framed in yellow. **(d)** Unsuccessful `Temporal verification` result. The elements causing not controllable executions are framed in pink.

configurations that cannot be executed without errors. TimeAwareBPMN-js helps understand such an interplay, showing possible inconsistencies.

- **Framework for new temporal verifying algorithms.** In the field of time-aware process models, different proposals deal with the representation and checking of temporal constraints and properties on BPMN models. Many proposals require converting a BPMN model into other formats and checking them using external tools like LoLA [34] or UPPAAL [24].
  TimeAwareBPMN-js architecture aims to integrate such transformations and external tools executions as plug-ins of the application. Such plug-ins can be easily built and integrated thanks to an application API. Using plug-ins, the user can focus on the logical aspects of the model without worrying about issues stemming from the translation of BPMN into other formalisms or the connection with external tools.
- **Applications.** In the BPM field, there is a significant interest in considering temporal aspects in workflow systems [2,3,12–14,17]. A critical aspect of such proposals is the lack of an actual application that allows one to use such time-aware (BPMN) models. This application would contribute to the spread of time-aware BPMN models, offering a solid and extensible framework to verify time-aware BPMN processes.

In future work, we plan to extend TimeAwareBPMN-js to ease the editing of temporal constraints, to represent fixed-date constraints, temporal constraints in other BPMN elements—such as loops, boundary events, and sub-processes—and give more details about the possible inconsistencies that can arise in a process.

Declaration of competing interest

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

Acknowledgments

We would like to thank the anonymous reviewers, and Prof. Carlo Combi and Prof. Barbara Oliboni from the Department of Computer Science of the University of Verona for their comments on the manuscript.

Funding

This work was partially supported by the INdAM–GNCS Project 2020 as part of project 2020 “Automated Reasoning about Time in Medical and Business Applications”.

References

[1] Object Management Group. Business process model and notation version 2.0.2. 2014, [Accessed 7 July 2021]. https://www.omg.org/spec/BPMN/2.0.2/PDF.
[2] Eder J, Panagos E, Rabinovich M. Workflow time management revisited. In: Seminal contributions to information systems engineering: 25 years of CAiSE. Springer; 2013, p. 207–213. http://dx.doi.org/10.1007/978-3-642-36926-1_16.
[3] Cheikhrouhou S, Kallel S, Guermouche N, Jmaiel M. The temporal perspective in business process modeling: A survey and research challenges. Serv Oriented Comput Appl 2015;9(1):75–85. http://dx.doi.org/10.1007/s11761-014-0170-x.
[4] Gagne D, Trudel A. Time-BPMN. In: 2009 IEEE conference on commerce and enterprise computing. 2009, p. 361–367. http://dx.doi.org/10.1109/CEC.2009.71.
[5] Combi C, Oliboni B, Zerbato F. A modular approach to the specification and management of time duration constraints in BPMN. Inf Syst 2019;84. 111–144. http://dx.doi.org/10.1016/j.is.2019.04.010.
[6] Lanz A, Weber B, Reichert M. Time patterns for process-aware information systems. Requir Eng 2014;19(2). 113–141. http://dx.doi.org/10.1007/s00766-012-0162-3, orgname = Springer.
[7] Cheikhrouhou S, Kallel S, Guermouche N, Jmaiel M. Enhancing formal specification and verification of temporal constraints in business processes. In: 2014 IEEE International Conference on Services Computing. 2014, p. 701–708. http://dx.doi.org/10.1109/SCC.2014.97.

[8] Watahiki K, Ishikawa F, Hiraishi K. Formal verification of business processes with temporal and resource constraints. In: 2011 IEEE international conference on systems, man, and cybernetics. 2011, p. 1173–118. http://dx.doi.org/10.1109/ICSMC.2011.6083857.

[9] Huai W, Liu X, Sun H. Towards trustworthy composite service through business process model verification. In: 2010 7th international conference on ubiquitous intelligence computing and 7th international conference on autonomic trusted computing. 2010, p. 422–427. http://dx.doi.org/10.1109/UIC-ATC.2010.114.

[10] Combi C, Oliboni B, Zerbato F. Modeling and handling duration constraints in BPMN 2.0. In: Proceedings of the Symposium on Applied Computing, SAC '17. ACM; 2017, p. 727–734. http://dx.doi.org/10.1145/3019612.3019618.

[11] Combi C, Posenato R. Controllability in temporal conceptual workflow schemata. In: Business Process Management, 7th International Conference, BPM 2009. LNCS, vol. 5701, Springer; 2009, p. 64–79. http://dx.doi.org/10.1007/978-3-642-03848-8_6.

[12] Lanz A, Posenato R, Combi C, Reichert M. Controlling time-awareness in modularized processes. In: Enterprise, Business-Process and Information Systems Modeling. BPMDS 2016, EMMSAD 2016. Lecture Notes in Business Information Processing, Springer; 2016, p. 157–172. http://dx.doi.org/10.1007/978-3-319-39429-9_11.

[13] Posenato R, Zerbato F, Combi C. Managing decision tasks and events in time-aware business process models. In: Business Process Management. BPM 2018.. 2018, p. 102–118. http://dx.doi.org/10.1007/978-3-319-98648-7_7, orgname = Springer, series=LNCS, volume=11080, isbn=978-3-319-98648-7.

[14] Posenato R, Lanz A, Combi C, Reichert M. Managing time-awareness in modularized processes. Softw Syst Modeling 2019;18(2). 1135–1154. http://dx.doi.org/10.1007/s10270-017-0643-4.

[15] Franceschetti M, Eder J. Semi-contingent task durations: Characterization and controllability. In: International conference on advanced information systems engineering (CAISE). Springer; 2021, p. 246–261. http://dx.doi.org/10.1007/978-3-030-79382-1_15.

[16] Maggi FM, Montali M, Peñaloza R, Alman A. Extending temporal business constraints with uncertainty. In: Business Process Management. BPM 2020. 12168, Springer; 2020, p. 35–54. http://dx.doi.org/10.1007/978-3-030-58666-9_3.

[17] Lanz A, Reichert M, Weber B. Process time patterns: A formal foundation. Inf Syst 2016;57:38–68. http://dx.doi.org/10.1016/j.is.2015.10.002.

[18] Dechter R, Meiri I, Pearl J. Temporal constraint networks. Artificial Intelligence 1991;49(1):61–95. http://dx.doi.org/10.1016/0004-3702(91)90006-6.

[19] Schwalb E, Vila L. Temporal constraints: A survey. Constraints 1998;3(2). 129–149. http://dx.doi.org/10.1023/A:1009717525330.

[20] Cheikhrouhou S, Kallel S, Guermouche N, Jmaiel M. Toward a time-centric modeling of business processes in BPMN 2.0.. In: International conference on information integration and web-based applications & services. ACM; 2013, p. 154–163. http://dx.doi.org/10.1145/2539150.2539182.

[21] Hunsberger L, Posenato R. Sound-and-complete algorithms for checking the dynamic controllability of conditional simple temporal networks with uncertainty. In: 25th International symposium on temporal representation and reasoning (TIME 2018). LIPIcs, vol. 120, 2018, p. 14:1–14:17. http://dx.doi.org/10.4230/LIPIcs.TIME.2018.14.

[22] Morales LEM, Tuñón MIC, Pérez MA. A formalization proposal of timed BPMN for compositional verification of business processes. In: Enterprise information systems. Lecture notes in business information processing, Springer; 2011, p. 388–403. http://dx.doi.org/10.1007/978-3-642-19802-1_27.

[23] Capel MI, Mendoza LE. Automating the transformation from BPMN models to CSP+t specifications. In: 2012 35th annual IEEE software engineering workshop. 2012, p. 100–109. http://dx.doi.org/10.1109/SEW.2012.17.

[24] Bengtsson J, Larsen K, Larsson F, Pettersson P, Yi W. UPPAAL—a tool suite for automatic verification of real-time systems. In: Hybrid Systems III. Springer; 1996, p. 232–243. http://dx.doi.org/10.1007/BFb0020949.

[25] ATAPIS-adaptive time- and process-aware information systems. 2019, URL https://www.uni-ulm.de/en/atapis.

[26] Goser K, Jurisch M, Acker H, Kreher U, Lauer M, Rinderle-Ma S, et al. Next-generation process management with ADEPT2. In: Proceedings of the bpm demonstration program at the fifth international conference on business process management. 2007, p. 4.

[27] Durán F, Rocha C, Salaün G. Stochastic analysis of BPMN with time in rewriting logic. Sci Comput Program 2018;168:1–17. http://dx.doi.org/10.1016/j.scico.2018.08.007.

[28] Houhou S, Baarir S, Poizat P, Quéinnec P. A direct formal semantics for BPMN time-related constructs. In: Proceedings of the 16th international conference on evaluation of novel approaches to software engineering. SCITEPRESS; 2021, p. 138–149. http://dx.doi.org/10.5220/0010462901380149.

[29] Jackson D. Software abstractions: logic, language, and analysis. MIT Press; 2012.

[30] Cheikhrouhou S, Kallel S, Jmaiel M. Toward a verification of time-centric business process models. In: 2014 IEEE 23rd international WETICE conference. 2014, p. 326–331. http://dx.doi.org/10.1109/WETICE.2014.75, issn=1524-4547.

[31] Camunda. bpmn-js, BPMN 2.0 viewer and editor. 2014, [Accessed 7 Jul. 2021], https://bpmn.io/toolkit/bpmn-js/.

[32] Posenato R. CSTNU Tool: A java library for checking temporal networks. SoftwareX 2022;17:100905. http://dx.doi.org/10.1016/j.softx.2021.100905.

[33] Zavatteri M, Viganò L. Conditional simple temporal networks with uncertainty and decisions. Theoret Comput Sci 2018. http://dx.doi.org/10.1016/j.tcs.2018.09.023.

[34] Schmidt K. LoLA a low level analyser. In: Application and Theory of Petri Nets 2000. Springer; 2000, p. 465–474. http://dx.doi.org/10.1007/3-540-44988-4_27.