Managing Decision Tasks and Events in Time-Aware Business Process Models

Roberto Posenato[(✉)], Francesca Zerbato, and Carlo Combi

Dipartimento di Informatica, Università degli Studi di Verona, Verona, Italy
`roberto.posenato@univr.it`

Abstract. Time-aware business process models capture processes where temporal properties and constraints have to be suitably managed to achieve proper completion. Temporal aspects also constrain how decisions are made in processes: while some constraints hold only along certain paths, decision outcomes may be restricted to satisfy temporal constraints. In this paper, we present *time-aware* BPMN processes and discuss how to: (i) add temporal features to process elements, by considering also the impact of events on temporal constraint management; (ii) characterize decisions based on when they are made and used within a process; (iii) specify and use two novel kinds of decisions based on how their outcomes are managed; (iv) deal with intertwined temporal and decision aspects of time-aware BPMN processes to ensure proper execution.

1 Introduction and Motivation

Time-awareness is undeniably a crucial property of business processes [13,24]. In the last years, temporal features of process models have been widely considered and studied with a focus on different intertwined aspects. Among them, we mention the modeling and checking of temporal constraints at design time [6, 13,17,23], the management of uncertainty for task duration [9], the modular design of time-aware processes [24], the specification of time patterns [20], and the modeling of temporal constraints in Business Process Model and Notation (BPMN) [5,8,12].

In general, temporal features of process models have to be dealt with by considering how they relate to the semantics of process elements. Particularly, decision tasks and events [22] are important concepts to consider jointly with temporal constraints, as they represent points in the process flow where information is acquired and used to determine the following flow of process execution. Indeed, information about decisions is used by exclusive gateways to choose one among alternative execution flows, based on the evaluation of conditions previously set by decision tasks or related to event occurrence.

Thus, during execution time-aware processes have to face two different kinds of uncertainty, one related to activity duration, which is known only after activity completion, the other one stemming from the outcomes of decision tasks and

from events that determine which process path to follow. Such uncertainties are solved only when tasks have been executed and events have occurred.

At design time, given a time-aware business process model, it is desirable to know whether it is possible to execute it in a correct way, by considering all the possible combinations of activity durations and decision outcomes. However, such durations and outcomes are not under the control of a process engine. In this scenario, if an engine can plan the execution of future steps considering only the history of already executed elements and made decisions, and guaranteeing that all the specified temporal constraints are satisfied, we say that the process is *dynamically controllable*.

In this paper, we propose a new *time-aware* well-structured process model based on BPMN [22] to handle the subtle relations between temporal constraints and decisions and show how to check if process cases can be executed successfully. The main novelties of our approach can be summarized as follows: (i) we add temporal features to process elements, considering also the impact of event occurrence on temporal constraint management, (ii) we discuss the relation between the making and the use of decisions, (iii) we conceptually distinguish two novel types of decisions, (iv) we describe how to deal with both the uncertainty related to the effective duration of executed activities and the uncertainty related to decisions made during the process execution, (v) we define the notion of *dynamic controllability (DC)* for such processes, and (vi) we show a mapping of time-aware well-structured BPMN processes onto suitable temporal constraint networks to check the dynamic controllability of such processes.

1.1   A Motivating Example Taken from the Clinical Domain

As a motivating scenario, let us consider the management of patients diagnosed with knee osteoarthritis (OA). The process diagram of Fig. 1 shows some important treatment steps, excerpt from widely adopted clinical practice guidelines [2]. The core of the diagram is designed by using BPMN [22], which is enriched with different kinds of temporal constraints, such as activity/gateway durations, event waiting times, sequence flow delays, and relative temporal constraints.

Knee OA is a common degenerative joint disease involving cartilage and nearby tissues [2].

In Fig. 1 we focus on pharmacologic treatment, thus leaving nonpharmacologic treatment represented as collapsed subprocess NonPhTr. Moreover, we specify a type only for decision tasks: those representing human decision-making are given type *user*, while tasks enclosing a detailed decision logic are of type *business rule*.

Prior to prescribing treatments, a physician in charge must *Check Contraindications* (business rule task T_0) to commonly administered drugs, such as paracetamol, Non-Steroidal Anti-Inflammatory Drugs (NSAIDs), and opioids. Being potentially life-threatening, absolute contraindications are precisely defined in clinical guidelines to avoid misinterpretation (hence the use of a business rule task).
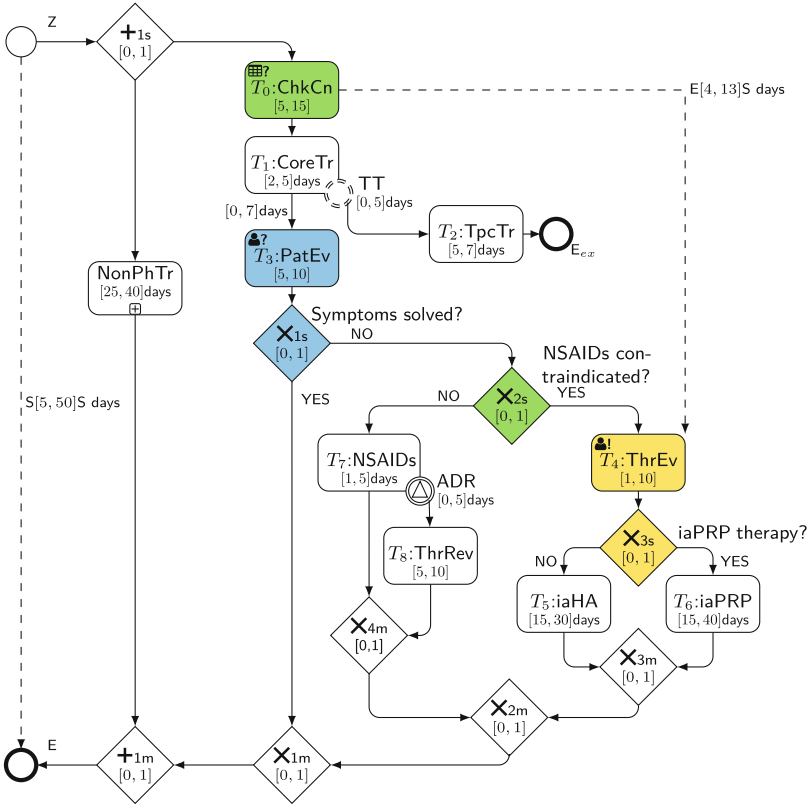
Fig. 1. BPMN process diagram showing the main steps for treating knee osteoarthritis. If not specified, the granularity of temporal ranges in the diagrams is minute.

Depending on the assessed drug tolerance, different treatments are prescribed, in a stepwise manner. The *Core Treatment* (task T_1) is essentially based on paracetamol. To improve pain control, topical preparations may be added during core treatment. In Fig. 1 the need for topical treatment is represented by non-interrupting event TT, which leads to the *Topical Treatment* itself (task T_2). However, paracetamol is often insufficient, even if combined with topical treatment. Thus, a *Patient Evaluation* (task T_3) is scheduled afterwards. If symptoms persist (gateway $\times_{1s}$), an advanced treatment must be prescribed. The choice (gateway $\times_{2s}$) of the best treatment for the patient depends on the contraindications evaluated in T_0. In case of contraindications, intra-articular therapy is preferred. Among existing alternatives, intra-articular hyaluronic acid (iaHA) and platelets-rich plasma (iaPRP) have shown similar efficacy [16]. Therefore, during *Therapy Evaluation* (user task T_4) the physician can choose the therapy among the available ones. This decision differs from those made in T_0 and T_3 since not all of the possible outcomes are required to be available at runtime to guarantee that the process can be executed successfully. Indeed, it is

sufficient that at least one of the available therapies iaHA and iaPRP can be chosen. Then, gateway $\times_{3s}$ uses the decision made in T_4 to route the process flow towards either T_5 or T_6. If there are no contraindications, NSAIDs drugs (task T_7) may be administered. However, severe adverse drug reactions (ADRs) may occur while taking NSAIDs: if an ADR is reported, the treatment is immediately interrupted. In Fig. 1, this scenario is captured by signal boundary event ADR, whose occurrence interrupts task T_7 and leads to *Therapy Re-evaluation* (task T_8).

The example shows how process execution relies also on temporal and decision aspects. On the one hand temporal constraints must be satisfied to guarantee that the process is completed successfully. On the other hand, decision tasks determine which process path is preferred over another.

2   Characterization of Time-Aware BPMN Processes

In this section, we introduce temporal aspects, distinguish the two types of decisions that characterize the novel time-aware BPMN and discuss their relations. Then, we propose the notion of dynamic controllability for time-aware BPMN processes. Hereinafter, we consider only well-structured processes as they offer several advantages in terms of comprehension, modularity, and robustness [6,11, for a detailed discussion].

2.1   Specification of Temporal Properties and Constraints

Here, we enrich BPMN processes by adding a temporal dimension to a relevant subset of BPMN elements and suitable temporal constraints based on concepts presented in [6,9,23]. The obtained *time-aware* BPMN fosters the temporal characterization of tasks, gateways, events, sequence flow edges, and time lags between process elements. We describe the introduced temporal aspects by referring to the example of Fig. 1 and borrowing the notions of "activity/event activation" and "event triggering/handling" from the BPMN standard [22].

– **Activities** have a duration attribute represented as a range $[x,y]G$ with $0 < x < y < \infty$, where x/y is the minimum/maximum allowed time span for an activity to go from state "started" to "completed" [20] and G (Granularity) stands for the time unit used (e.g., seconds,...). At run-time, the real duration of an activity cannot be fixed by the process engine, but only observed after who is in charge of executing it completes the activity (*contingent duration*). Process engine takes into account the real duration to properly enact the following elements. Who is in charge of executing the activity must observe the two bounds x and y.
  For example, T_3 has a duration $[5,10]$ *min*: physicians in charge of T_3 may take between 5 and 10 min to execute it.
– **Intermediate catching events** have a temporal property, $[x,y]G$ with $0 \leq x \leq y < \infty$, representing the minimum/maximum amount of time during

which they may be triggered (*event waiting time*). When x is 0, it means that the event may be triggered as soon as it is activated. y is the upper bound on the amount of time allowed for event triggering that prevents the process to wait infinitely for the occurrence of an event. Since the triggering of an event is not controlled by the process engine, the actual event waiting time is only known at run-time. When a catching event is attached to the boundary of an activity, its waiting time is implied by the duration of the activity. Specifically, if activity T has a duration $[x, y]mG$ and a boundary event e, then the event waiting time for e must be $[0, y']mG$ where $y' < y$ and mG is the minimum time granularity considered in the process. This ensures that e cannot occur at the T completion instant as required by [22]. For practicality, in case of coarser granularity the model admits $y' \leq y$ assuming that y' is always before y after the conversion to the minimum granularity. As an example, task T_7, having duration $[1, 5]$ days, and boundary event ADR, having waiting time $[0, 5]$ days: in this case, since days are coarser than minutes (the minimum granularity), the upper bounds coincide. If e is *non-interrupting* (e.g. TT in Fig. 1), BPMN requires that the duration of the associated activity includes the duration of all non-interrupting event handlers [22]. As discussed later, such specification can be satisfied by combining the described temporal properties, thus allowing designers to think about the elementary temporal characterization of activities.

– **Gateways and sequence flow** also have a duration range of the form $[l, u]$ G, with $0 \leq l \leq u \leq \infty$. However, in this case, the process engine plans the real execution time for such elements by choosing a suitable value of the range. A range associated to a sequence flow edge connecting an element A to an element B is called *sequence flow delay* because it represents the possibility for the process engine to delay the enacting of B after A is completed. For example, between T_1 and T_3 there is a delay of $[0, 7]$ days: considering the decision in T_0 and the completion time of T_1, the engine could reduce this delay even to 0 to guarantee that following tasks can be executed without constraint violations. If a designer does not set a duration, it is assumed to be $[0, \infty]mG$.

– **Relative constraints** are depicted in Fig. 1 as dashed edges that connect any two process nodes [9]. Relative constraints limit the time distance between the starting/ending instants of two elements and have the form $I_S[u, v]I_F G$, where I_S is the starting (S)/ending (E) instant of the first element, while I_F is the starting/ending instant of the second one [9]. For example, the time distance between the end of task T_0 and the beginning of task T_4 is given by E[4, 13]S days meaning that, if iaHA or iaPRP are needed, the decision of which one to prescribe must be made after at least 4 days and before 13 days from the completion of T_0. To deal with event instantaneity, we choose to always adopt the notation I_S to denote the triggering instant of one event. For example, constraint S[5, 50]S days represents the overall minimum and maximum process durations and holds between events Z and E.

2.2   Specification of Decisions: A Novel View on Decision Outcomes

The modeling decisions associated to processes is becoming increasingly important. Here, we offer a novel view on decision tasks, based on where and how their outcomes are used in a process.

In our proposal, decisions are made in decision tasks and any following exclusive gateway may use the outcome of such decisions to route the process flow [22]. In this way, there is a greater flexibility compared to the assumption that decisions are made by a decision task immediately preceding the exclusive gateway [1]. Moreover, allowing a decision to be made at any place prior to an exclusive gateway, may increase temporal flexibility during process execution. For example, T_0 determines the therapies contraindicated for a certain patient, thus affecting which process path is taken at gateway $\times_{2s}$. If the outcome of T_0 is that NSAIDs are contraindicated then, to guarantee that at least one of T_5 and T_6 can be chosen, the delay $[0,7]$days between T_1 and T_3 must be set to 3 days at the most, for not precluding the possibility to satisfy also the overall duration constraint $S[5,50]S$ days. Otherwise, a delay of 7 days can be allowed. Decision tasks and corresponding gateways are given the same coloring scheme to highlight the connection between where decisions are made and where they are used. W.l.o.g, we assume that exclusive gateways are binary, i.e., they only have two alternative outgoing sequence flows. Indeed, if a decision has n alternative outcomes, these can be evaluated by setting a proper sequence of $\lceil \log n \rceil$ exclusive gateways.

Beside decision tasks and exclusive gateways, interrupting boundary events, such as ADR of Fig. 1, may also represent decisions as their occurrence determines the enactment of an alternative, exception flow. However, being their triggering instantaneous, interrupting boundary events always represent points in the process where a decision is made and used at the same time.

Beside their position, decisions may be distinguished at a conceptual level based on the availability of their outcomes during process run-time. In general, a process should be guaranteed to be executable for any possible combination of decision outcomes also with respect to the temporal constraints. Since such requirement can be very strict, sometimes it reasonable to relax it by admitting that some paths are not executable at run-time if temporal constraints cannot be satisfied. In other words, it is reasonable to reduce the possible outcomes of some decisions in order to guarantee that the process can be completed successfully.

In this regard, we propose a novel perspective aimed to conceptually distinguish decisions based on how their outcomes may be chosen at run-time.

Some decisions represent the response of the process to conditions that are dictated by the context in which the process is executed, such as data-based conditions or event occurrence. At run-time the process must always be guaranteed to run **any** of such alternative outcomes. We refer to decisions of this kind as *observations*: *A decision is called* observation *when the number of its possible outcomes cannot be reduced at run-time.* Task T_0 makes an observation: Physicians must determine which drugs are contraindicated based on well-documented

evidence. For every possible outcome (alternative flows in $\times_{2s}$), the process must be executable for any possible duration of other process tasks.

Conversely, for some decisions it is possible to limit their outcomes at run-time if this can help to execute the process successfully. In this case, the choice of the outcomes is still arbitrary, but the set of possible outcomes can be reduced considering the past execution of previous elements. It must be ensured that **at least one** outcome is always allowed. To denote that the decision is guided by the limitation of its outcomes we refer to decisions of this kind as *guided decisions*: *A decision is a* guided decision *when its possible outcomes can be reduced at run-time to comply with temporal constraints.* In Fig. 1, T_4 makes a guided decision. Since iaHA and iaPRP have similar efficacy and safety, physicians may suggest one or the other without the need for both to always be available when the decision is made. At run-time, if T_4 has been enacted 13 days after T_0 (in case that TT is triggered at day 5 and T_2 lasts 7 days), then T_6 cannot be allowed as its execution could violate the upper bound of the process duration constraint S[5, 50]S; therefore, T_4 can only select iaHA task.

In Fig. 1, symbol **?** next to the task type icon denotes decision tasks that make observations, and symbol **!** denotes tasks that make guided decisions. Decisions based on the occurrence of boundary events are always *observations*.

2.3   Controllability of a Time-Aware BPMN Process

From a temporal perspective, *executing* a time-aware BPMN process P means: (i) to schedule the starting time of all elements, (ii) to set the duration of gateways and sequence flow delays, and (iii) to determine which are the allowed outcomes of a guided decision before enacting the corresponding decision task. The values of observations in P are not known in advance as they are incrementally revealed over time as decision tasks are executed. Similarly, the durations of activities are only known as the activities complete. Therefore, a *dynamic* execution of P must *react* to observations and contingent durations in real time. A *viable* execution is one that guarantees that all *relevant* constraints–those holding in the paths being executed–will be satisfied no matter which observation outcomes and durations are revealed over time. A time-aware BPMN process with a dynamic and viable strategy is called *dynamically controllable* (DC).

3   Dynamic Controllability Checking

In this section, we show how to determine if a time-aware BPMN schema is DC. First, we introduce a temporal-constraint model, called Conditional Simple Temporal Network with Uncertainty and Decision (CSTNUD) [26], that results to be a well-founded model for representing and reasoning about temporal constraints; then, we show how to verify the dynamic controllability of a process model P using a corresponding CSTNUD S_P.

3.1 A Short Introduction to CSTNUD

In general, a temporal-constraint network can be viewed as a graph in which nodes represent real-valued variables and edges represent binary constraints on variables. The kind of binary constraint that can be attached to edges characterizes the network and its expressive power. For example, in a *Simple Temporal Network* (STN) [10] $(\mathcal{T}, \mathcal{C})$, where $\mathcal{T}$ is a set of real-valued variables, called *time-points,* and $\mathcal{C}$ is a set of binary constraints, each constraint has the form $(Y - X \leq \delta)$, where $X, Y \in \mathcal{T}$ and $\delta \in \mathbb{R}$. When it is possible to assign a value to each time-point of a STN such that all constraints are satisfied, then the STN is said to be *consistent.*

By *executing* a temporal-constraint network we mean that the assignment of values to time-points is made by an (executing) engine incrementally following an execution strategy. An execution strategy determines the schedule to apply. For example, if an STN S has a solution, the *earliest execution strategy* schedules S assigning to each time-point its earliest possible execution time.

In [7], Hunsberger et al. proposed Conditional Temporal Network with Uncertainty (CSTNU), that extends STNs including *scenarios* [14] and *contingent links* [21]. A scenario specifies which time-points and constraints to consider during an execution and it is represented by a conjunction of propositional literals. The value of each proposition is unveiled during the execution (environment decides its value). A contingent link is a special kind of temporal constraint having the form, (A, x, y, C), where A and C are time-points, and $0 < x < y < \infty$. Typically, it assumed that a contingent link is activated when A is executed. Then, the value for setting C is decided by the environment not by the engine. However, C is guaranteed to execute such that the temporal difference, $C - A$, is between x and y, i.e., the contingent link is satisfied. Contingent links are used to represent actions with uncertain durations.

In [3] the authors propose CSTN with Decisions (CSTND), a generalization of STN with scenarios (CSTN) that allows some of the propositional variables to be assigned not by the environment, but by the engine executing the network. In [26] CSTND are generalized incorporating contingent links. The resulting network is called a CSTNU with Decisions (CSTNUD).

In the following we combine and extend concepts from earlier work [3,7,14, 26].

Definition 1 (Label representing scenario). *Let $\mathcal{P}$ be a set of propositional letters. A* label *ℓ over $\mathcal{P}$ is a conjunction, $\ell = l_1 \wedge \cdots \wedge l_k$, of literals $l_i \in \{p_i, \neg p_i\}$ on distinct variables $p_i \in \mathcal{P}$. The empty label is denoted by $\boxdot$. For labels, ℓ_1 and ℓ_2, if $\ell_1 \models \ell_2$, we say ℓ_1 entails ℓ_2. $\mathcal{P}^*$ denotes the set of all labels over $\mathcal{P}$.*

Definition 2 (CSTNUD). *A* Conditional STN with Uncertainty and Decision *is a tuple, $\langle \mathcal{T}, \mathcal{P}, \mathcal{CP}, \mathcal{DP}, \mathcal{C}, \mathcal{OT}, \mathcal{O}, \mathcal{L} \rangle$, where:*

- *$\mathcal{T}$ is a finite set of* temporal variables *or* time-points*;*
- *$\mathcal{P}$ is a finite set of* propositional variables, *i.e., boolean variables;*
- *$(\mathcal{CP}, \mathcal{DP})$ is a partition of $\mathcal{P}$ into* contingent propositional variables (**observations**) *$\mathcal{CP}$ and* controllable propositional variables (**decisions**) *$\mathcal{DP}$;*
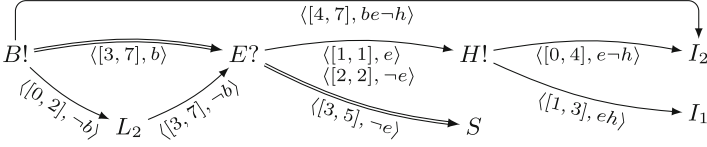
Fig. 2. An example of CSTNUD. b, e and h are relative to $B!, E?$ and $H!$, respectively.

- $\mathcal{C}$ *is a finite set of* labeled constraints, *each of the form,* $(l \leq Y - X \leq u, \ell)$, *where:* $X, Y \in \mathcal{T}$; $l \leq u$; $l, u \in \mathbb{R}$; *and* $\ell \in \mathcal{P}^*$;
- $\mathcal{OT} \subseteq \mathcal{T}$ *is the set of* disclosing time-points; *and*
- $\mathcal{O}: \mathcal{P} \rightarrow \mathcal{OT}$ *is a bijection that associates each* $p \in \mathcal{P}$ *to a disclosing time-point* $\mathcal{O}(p) \in \mathcal{OT}$, *i.e., to a time-point that, when executed, determines the disclosure of value of the associated proposition variable. If* $p \in \mathcal{CP}$, *then its* $\mathcal{O}(p)$ *is called an* observation time-point; *otherwise a* decision time-point.
- $\mathcal{L}$ *is a set of* contingent links *each of the form* (A, x, y, C, ℓ), *where:* $0 < x < y < \infty$; $A, C \in \mathcal{T}$ *are the* activation *and* contingent *time-points;* $\ell \in \mathcal{P}^*$; *and distinct contingent links have distinct contingent time-points.*

When an observation time-point is executed, the environment assigns a truth value to the corresponding observation; however, when a decision time-point is executed, the decision is assigned a truth value by the engine. $P!$ represents the decision time-point associated to decision p, while $Q?$ the observation time-point associated to observation q. As shown in [4], w.l.o.g. in Definition 2 only constraints are labeled, not time-points.

Viewed as a graph, a CSTNUD edge represents either a labeled constraint or a contingent link. In particular, each edge having the form $X \xrightarrow{\langle [l, u], \ell \rangle} Y$ represents a labeled constraint, $(l \leq Y - X \leq u, \ell)$, and it is called also *standard edge*; each edge having the form $A \xrightarrow{\langle [x, y], \ell \rangle} C$ represents the labeled contingent link (A, x, y, C, ℓ), and it is called *contingent*. The pair $\langle [l, u], \ell \rangle$ is called *labeled range/value*. If between two time-points there exist more labeled constraints, the standard edge connecting them has more labeled ranges, one for each labeled constraint.

Figure 2 shows a CSTNUD having 7 time-points of which 2 are contingents and 3 are disclosing time-points. Contingent link $(B!, 3, 7, E!, b)$ is activated only if decision b is true, while contingent link $(E?, 3, 5, S, \neg e)$ is activated only if observation e is false. For example, if $(B!, 3, 7, E!, b)$ is executed (b true), and it lasts 7, and the observation e results true, for executing the network without violating the constraint between $B!$ and I_2 is necessary to set decision h to false.

In [26], the execution semantics of a CSTNUD is given as a two-player game in which $\mathtt{Pl}_1$ models the executing agent and $\mathtt{Pl}_2$ models the environment, assumed as the most powerful possible player. A game runs in turns: at any time instant t, there exist two turns: the $\mathtt{Pl}_1$ turn, $T_1(t)$, and the $\mathtt{Pl}_2$ one, $T_2(t)$, occurring after $T_1(t)$. At each turn, a player may decide to make k move(s), with $0 \leq k < \infty$. A $\mathtt{Pl}_1$ move is either the execution of a non-contingent time-point X or the assignment of a truth-value to a decision d. A $\mathtt{Pl}_2$ move is either the execution

of a contingent time-point C or the assignment of a truth-value to a observation p. $\mathtt{Pl_2}$ is guaranteed to always have full information on what $\mathtt{Pl_1}$ has done before. During the game, the conjunction of truth-values of propositional variables is represented by the label ℓ_{cps}. $\mathtt{Pl_1}$ wins the game when there are no more time-points to execute and for each constraint $(l \leq Y - X \leq u, \ell) \in \mathcal{C}$ such that $\ell_{cps} \models \ell$, then the execution times $S(X)$ of X and $S(Y)$ of Y satisfy the constraint $l \leq S(Y) - S(X) \leq u$. $\mathtt{Pl_2}$ wins otherwise. We denote by σ_i a winning strategy if $\mathtt{Pl_i}$ wins the game by following σ_i. Informally, a CSTNUD has the dynamic controllability property if $\mathtt{Pl_1}$ has a winning strategy that is based only on the history of past moves made in the game. The history is defined in terms of *execution sequence*, the ordered sequence of executed time-points and assigned propositions [26]. Usually, $Z_1(Z_2)$ represents an execution sequence of $\mathtt{Pl_1}$ ($\mathtt{Pl_2}$) and $\sigma_1(Z_1, t)(\sigma_2(Z_2, t))$ represents a moved-based strategy that tells a player to make a move at time instant t only if the move is applicable at t [26].

Definition 3 (Dynamic Controllability [26]**).** *A CSTNUD is* dynamically controllable (DC) *if $\mathtt{Pl_1}$ has a winning strategy such that for any $t > 0$ and any pair of execution sequences Z_1, Z_2, if $\sigma_2(Z_1, t') = \sigma_2(Z_2, t')$ for $0 \leq t' < t$, then $\sigma_1(Z_1, t) = \sigma_1(Z_2, t)$.*

In [26] a DC checking algorithm for CSTNUDs based on Timed Game Automata (TGA) is proposed, while a DC checking algorithm based on constraint propagation for a sub class of CSTNUDs is presented in [3].

3.2   Mapping Time-Aware BPMN onto CSTNUD

To verify the dynamic controllability of a process model P, it is convenient to transform it into an CSTNUD S_P using the transformation rules depicted in Tables 1 and 2. Such rules are described in the proof of Theorem 1. The obtained S_P may be checked for DC by applying one of the available algorithms for DC checking [3, 26]. The following theorem shows that the process model P results to be DC if and only if S_P is DC.

Theorem 1. *Given a time-aware BPMN process P, there exists a CSTNUD S_P such that P is dynamically controllable if and only if S_P is DC.*

Proof. W.l.o.g., we assume that all temporal ranges have the same base granularity mG. In case that P contains ranges with different time granularities, it is possible to convert them to mG. Tables 1 and 2 give the mapping of the elements that can be used to transform a time-aware BPMN fragment into the corresponding CSTNUD. By applying the proposed mappings to P, one can simply verify that the obtained S_P represents all precedence relations and temporal constraints of P. Let us consider each mapping in detail.

– $\mathtt{Task/Subprocess}$. Each task A is transformed into two CSTNUD time-points, A_S and A_E, representing its start and end instants. The duration range, $[x, y]$, is converted to the contingent link (A_S, x, y, A_E, ℓ). The label ℓ is determined

Table 1. Mapping of time-aware BPMN fragments to CSTNUDs.

Time-aware BPMN fragment	Corresponding CSTNUD	Time-aware BPMN fragment	Corresponding CSTNUD
Start/End event	$Z \rightarrow$ $\rightarrow E$	**Task/Subprocess**	$\rightarrow A_S \xrightarrow{\langle [x,y],\ell \rangle} A_E \rightarrow$
Decision Task (observation)	$\rightarrow A_S \xrightarrow{\langle [x,y],\ell \rangle} A_E$ $\xleftarrow{\langle [0,0],\ell \rangle}$ $\leftarrow A_{\lceil \log n \rceil}? \leftarrow \cdots \leftarrow A_1?$	**Decision Task (guided decision)**	$\rightarrow A_S \xrightarrow{\langle [x,y],\ell \rangle} A_E$ $\xleftarrow{\langle [0,0],\ell \rangle}$ $\leftarrow A_{\lceil \log n \rceil}! \leftarrow \cdots \leftarrow A_1!$
AND Split	$\rightarrow +_S \xrightarrow{\langle [l,u],\ell \rangle} +_E$	**AND Join**	$+_S \xrightarrow{\langle [l,u],\ell \rangle} +_E \rightarrow$
XOR Split	$\rightarrow +_S \xrightarrow{\langle [l,u],\ell \rangle} +_E \xrightarrow{\langle [0,\infty],\ell p \rangle} \xrightarrow{\langle [0,\infty],\ell\neg p \rangle}$ associated to p proposition.	**XOR Join**	$\xrightarrow{\langle [0,\infty],\ell p \rangle} \xrightarrow{\langle [0,\infty],\ell\neg p \rangle} +_S \xrightarrow{\langle [l,u],\ell \rangle} +_E \rightarrow$ associated to p proposition.
Sequence flow delay $[l,u]$	$\langle [l,u],\ell \rangle$	**Intermediate Event** $I[x,y]$	$\rightarrow I_S \xrightarrow{\langle [x,y],\ell \rangle} I_E? \rightarrow$
Boundary Interrupting Event	$B_S \xrightarrow{\langle [0,0],\ell \rangle} A_S$ $\langle [0,y'],\ell b \rangle \; \langle [x,y],\ell b \rangle$ $B_E? \qquad A_E$ $\langle [0,\epsilon_2],\ell b \rangle$ $BP_S \xrightarrow{\langle [0,\infty],\ell\neg b \rangle}$ $\langle [w,k],\ell b \rangle$ $BP_E \xrightarrow{\langle [0,\epsilon_3],\ell b \rangle} \times \rightarrow$	**Boundary Non-Interrupting Event**	$B_S \xleftarrow{\langle [0,0],\ell \rangle} A_S$ $\langle [0,y'],\ell b \rangle \; \langle [x,y],\ell b \rangle$ $B_E? \qquad A_E$ $\langle [0,\epsilon_2],\ell b \rangle$ BP_S $\langle [w,k],\ell b \rangle$ $BP_E \xrightarrow{\langle [0,\infty],\ell b \rangle}$ $\langle [0,\infty],\ell b \rangle$ $E_{BP} \qquad \langle [0,\epsilon_1],\ell \rangle$

If not specified, the scenario is assumed to be ℓ and the temporal range of an edge is $\langle [0,\infty],\ell \rangle$.

considering the (possible) XORSplit/XORJoin gateways that are present in the path, Π, from the start event to A in P: (i) Initialize $\ell = \boxdot$; (ii) For each (possible) XORSplit $\times_i$ in Π associated to proposition p, add p or $\neg p$ to ℓ according to the branch present in Π; (iii) For each (possible) XORJoin $\times_i$ in Π associated to proposition p, remove any p literal from ℓ. The obtained label represents the scenario in S_P where A_S and A_E have to be executed and their contingent link observed. The mapping of a subprocess onto a CSTNUD is equivalent.

– **Decision Task (observation).** The conversion is analogous to the one of a task as for its duration attribute. As regards the observation made, it is necessary to represent all the possible outcomes by adding to S_P a suitable number of observation time-points. In particular, if an observation of a decision task A in P can assume n distinct values, then, in S_P there must be $\lceil \log n \rceil$ propositions, associated to new $A?_1, \ldots, A?_{\lceil \log n \rceil}$ observation time-points. In this way, each A outcome is represented by a proper combination

of truth-values of such $\lceil \log n \rceil$ propositions. $A?_1, \ldots, A?_{\lceil \log n \rceil}$ are added in sequence after the CSTNUD time-point A_E. The temporal distance between $A?_1, \ldots, A?_{\lceil \log n \rceil}$ and A_E is set to 0 to constrain that the observation values are available at the same instant in which A_E is executed.

- `Decision Task (guided decision)`. The conversion is analogous to the one of a decision task making an observation. In this case, however, the possible outcomes of a decision task are represented using decision time-points instead of observation ones to capture the semantics associated to guided decisions.
- `ANDSplit/ANDJoin` gateways. The conversion is analogous to the one of a task. In this case, however, duration attribute $[x, y]$ is converted to a standard edge as gateways are executed by the process engine.
- `XORSplit/XORJoin` gateways. The conversion is analogous to the one of an ANDSplit/ANDJoin as regards its duration attribute. As for scenario, if ℓ is the scenario in which a XORSplit is present, then all converted elements located in its *true* outgoing flow will have a label entailing ℓp, while all converted elements located in its *false* outgoing flow will have a label entailing $\ell \neg p$, where p is the proposition associated to the considered XORSplit. In case of XORJoin, the process is reverse: the scenario label is updated removing p literal.
- `Sequence Flow Delay`. A sequence flow edge having temporal delay $[l, u]$ is converted to a standard edge having $\langle [l, u], \ell \rangle$ as labeled range.
- `Intermediate Event`. Since the temporal range of an intermediate event represents the waiting time allowed for event triggering, its mapping is analogous to the one of a task. Moreover, the end time-point $I_E?$ is also an observation time-point associated to a proper proposition i for representing if the event occurred (true value). In this way, the semantics of the CSTNUD fragment is the following: the execution of $I_E?$ reveals if the event occurred or not and, in case of occurrence, the execution time of $I_E?$ is the exact instant in which the event occurred.
- `Boundary Interrupting Event`. This conversion can be split in two parts that work in parallel, one for task A and one for interrupting event B. $y' < y$ as required by BPMN [22]. Task A and event B are mapped using the previous mappings for tasks and intermediate events, respectively. The contingent links associated to A and B must start at the same time: in Table 1, B_S is at distance 0 from A_S. In S_P, after $B_E?$, there are time-points and constraints related to the temporal characterization of subprocess BP (representing exception handling) that are labeled by ℓb to represent the fact that they must be considered only in case B occurs. Both A_E and BP_E are connected to time-point ✖ that represents the original exclusive gateway ✖ assumed instantaneous for simplicity.

Since the value of an observation cannot be constrained in any way, the obtained CSTNUD fragment allows also the representation (and reasoning) of cases that are not possible in a real process run. For example, in the CSTNUD it is possible that (i) the contingent link associated to B lasts more than the contingent link associated to A, (ii) the proposition b is set true and, therefore, (iii)

all temporal constraints associated to the interruption branch must be observed. But this case cannot occur in a real run of P because all interrupting events must occur prior to task completion [22]. Therefore, the dynamic controllability of this CSTNUD fragment guarantees the dynamic controllability of the original fragment containing Boundary Interrupting Event even for execution cases that can never occur. On the other hand, since it is necessary to guarantee the controllability for any possible combination of task duration and event occurrence, all real cases are simpler cases of the real worst case in which A completes at its maximum and B occurs at the last possible instant and BP completes at its maximum, that it is captured by the this conversion. In case that there exist some relative constraints (explained below) involving the start/end of task BP or the end of task A, in their corresponding CSTNUD edges, labels have to be adjusted considering literals $b/\neg b$ to guarantee that the edges are considered in the right scenarios. For example, in the edge associated to a relative constraint involving the end instant of A, the label must contain also $\neg b$ because the constraint has to be considered only when A is not interrupted.

– **Boundary Non-Interrupting Event**. The conversion is analogous to the one of a boundary interrupting event. In this case, however, there is a small complication given by the fact that the BPMN semantics dictates that, in case of event occurrence, task A (cf. Table 1) must wait the completion of event handler represented by BP for reaching state "complete" [22]. Thus, to properly represent such temporal constraint in the CSTNUD, it is necessary to add a join time-point **+** after A_E. **+** has to be executed after A_E and BP_E in case the event occurred (proposition b is true) and immediately after A_E in case the event did not occur. Such behavior is ensured by associating two temporal ranges to the edge $A_E \rightarrow$ **+**: $\langle [0,\infty], \ell b \rangle$ represents the delay to observe in case the event occurred (b is true), while $\langle [0,0], \ell \rangle$ represents the 0 delay in case the event did not occur. After **+**, there is the delay $\langle [0,\epsilon_1], \ell \rangle$ corresponding to the sequence flow delay associated to the flow outgoing of A in the BPMN fragment.
– **Relative Constraints**. Let us consider a relative constraint $\langle I_F \rangle [l, u] \langle I_S \rangle$ between two elements A and B, where I_F and I_S represent the kind of instants to be considered, i.e., S or E. In Table 2, A and B are tasks for space reasons, but they can be any combination of tasks, subprocesses, gateways and events. A relative constraint is converted to a CSTNUD edge between the time-points associated to instants A_{I_F} and B_{I_S} with labeled range $\langle [l, u], \ell \rangle$. If ℓ_A/ℓ_B is the scenario where A/B are mapped, then ℓ must satisfy $\ell \models \ell_A \ell_B$, i.e., relative constraints can be defined only in consistent scenarios.

As introduced above, a time-aware BPMN process is dynamically controllable if it is possible to execute it by satisfying all relevant constraints while reacting in real time to (i) the observation values that occur, (ii) tasks/subprocess durations, and (iii) event occurrences.

According to the two-player semantics of CSTNUDs, a CSTNUD is dynamically controllable if it is possible to execute it in a way such that, no matter

Table 2. Mapping of relative constraints between BPMN elements to CSTNUDs. For brevity, only tasks are exemplified but it is possible to consider any pair of elements.

Tasks A and B can be in different paths but not in not-compatible branches.

how the execution of any contingent link turns out and any observations turns out (Pl_2 execution strategy), it is possible to set a sequence of decisions and to schedule all non-contingent time-points in real time (Pl_1 strategy) satisfying all relevant constraints.

Considering the provided mapping and, in particular, the mapping of process fragments containing boundary events, it is a matter of definitions to verify that, given a process diagram P and its corresponding CSTNUD S_P, the dynamic controllability in S_P implies the dynamic controllability in P and vice-versa. □

4   Discussion and Related Work

Our approach is general and can be extended to other BPMN elements. For example, delays can be easily applied also to message flows while the concepts of duration and relative constraints can be applied to loop activities with some conditions. Moreover, we could easily include absolute temporal constraints. Our proposal of decision modeling can also be used to represent inclusive gateways. It would simply require to extend the mapping shown in Table 1.

The adoption of the CSTNUD model seems to be the more promising for checking a time-aware BPMN process considering techniques different from TGA as done in other proposals [5,25]. Indeed, while in [26] the author proposed a first CSTNUD DC checking algorithm based on TGA (The decision problem of reachability-time games in TGA with at least two clock is in EXP [15]), in [3] the problem of checking the DC of a CSTNUD was proven to be PSPACE-complete. Moreover, the authors proposed more efficient algorithms for checking the DC of some subclasses of CSTNUDs and this efficiency seems to be preserved for more general CSTNUD instances [3].

A significant analysis of temporal constraint modeling for process-aware information systems is presented in [20], where language-independent time patterns are defined, formalized, and their relevance is empirically shown.

As for modeling temporal constraints by means of BPMN or related extensions, relevant proposals are presented in [5,8,12,23,25]. In [25], the authors proposed an extension of BPMN for representing activity/process durations and

resource constraints and show how to check if a process satisfies or not business requirements by using TGA. In [5] authors adopted the same verification approach for another extension of BPMN including more kinds of temporal constraints, while in [12] the authors proposed an encoding of timed business processes into the Maude language for automatically verifying properties of a simpler extension of BPMN where only task timeouts and sequence flow delays can be expressed. All these proposals lack of a formal characterization of decisions and do not address how such decisions and temporal constraints can be modeled in the corresponding TGA/Maude language, which is indeed an important challenge. Finally, they do not consider events and their temporal characterization. In [8] the authors presented a modular approach to express various nuances of activity duration by combing BPMN elements in suitable blocks aimed to enhance re-use. Despite duration violations are managed, the authors did not consider the formal verification of the proposed cases. In [23], authors devised a process model enriched with temporal conditions in the formulation of conditional constructs, in particular XOR-splits and loops, and provided the related notions of schedule and controllability. The idea that a XOR-split/loop can check a temporal condition is quite interesting but they do not consider the issue of verifying process cases containing such rich elements.

In [18] authors introduced controlled violations (based on relaxation variables) of activity durations and proposed an approach based on constraint satisfaction to determine the best schedule for process while minimizing the cost of violations. Their approach does not consider events and different kinds of decisions.

Finally, in [19] the authors discuss *user decisions* made in knowledge-intensive processes by comparing them to decisions based on history data which cannot be freely made. Despite sharing the same concept of "freedom to decide", the proposed guided decisions limit their outcomes at run-time only when it is necessary to guarantee a successful execution.

5   Conclusions

In this paper, we presented a novel extension of time-aware BPMN where events can be temporally characterized and decisions are distinguished into two types.

As regards temporal characterization, we proposed to distinguish between durations that can be limited by a process engine and durations that become known only at run-time. As regards decisions, we proposed that they can be made and used in different points in the process to allow a greater flexibility with respect to temporal constraints. Moreover, they can be of two kinds: observations, for which the process has to be guaranteed to run for all possible outcomes, and guided decisions, for which the number of possible outcomes can be reduced at run-time to ensure a successful execution.

At application level, it is important to have processes that can be executed reacting to already executed activities, event occurrences and past decisions; we formalized this property as dynamic controllability for time-aware BPMN

processes. For verifying whether a BPMN process is dynamically controllable, we propose to map it onto a corresponding CSTNUD instance, whose dynamic controllability is likely to be checked by algorithms beyond the common adopted TGA model checking approach.

References

1. Batoulis, K., Weske, M.: Soundness of decision-aware business processes. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNBIP, vol. 297, pp. 106–124. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65015-9_7
2. Bruyère, O., et al.: An algorithm recommendation for the management of knee osteoarthritis in Europe and internationally. Semin. Arthritis Rheum. **44**(3), 253–263 (2014). https://doi.org/10.1016/j.semarthrit.2014.05.014
3. Cairo, M., et al.: Incorporating decision nodes into conditional simple temporal networks. In: 24th International Symposium on Temporal Representation and Reasoning (TIME 2017), pp. 9:1–9:18 (2017). https://doi.org/10.4230/LIPIcs.TIME.2017.9
4. Cairo, M., Hunsberger, L., Posenato, R., Rizzi, R.: A streamlined model of conditional simple temporal networks. In: 24th International Symposium on Temporal Representation and Reasoning (TIME 2017), vol. 90, pp. 10:1–10:19 (2017). https://doi.org/10.4230/LIPIcs.TIME.2017.10
5. Cheikhrouhou, S., Kallel, S., Guermouche, N., Jmaiel, M.: Enhancing formal specification and verification of temporal constraints in business processes. In: IEEE International Conference on Services Computing (SCC 2014), pp. 701–708 (2014). https://doi.org/10.1109/SCC.2014.97
6. Combi, C., Gambini, M., Migliorini, S., Posenato, R.: Representing business processes through a temporal data-centric workflow modeling language: an application to the management of clinical pathways. IEEE Trans. Syst. Man Cybern.: Syst. **44**(9), 1182–1203 (2014). https://doi.org/10.1109/TSMC.2014.2300055
7. Combi, C., Hunsberger, L., Posenato, R.: An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty - revisited. In: Filipe, J., Fred, A. (eds.) ICAART 2013. CCIS, vol. 449, pp. 314–331. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44440-5_19
8. Combi, C., Oliboni, B., Zerbato, F.: Modeling and handling duration constraints in BPMN 2.0. In: 32nd ACM Symposium on Applied Computing (SAC 2017), pp. 727–734 (2017). https://doi.org/10.1145/3019612.3019618
9. Combi, C., Posenato, R.: Controllability in temporal conceptual workflow schemata. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 64–79. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03848-8_6
10. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artif. Intell. **49**(1–3), 61–95 (1991). https://doi.org/10.1016/0004-3702(91)90006-6
11. Dumas, M., García-Bañuelos, L., Polyvyanyy, A.: Unraveling unstructured process models. In: Mendling, J., Weidlich, M., Weske, M. (eds.) BPMN 2010. LNBIP, vol. 67, pp. 1–7. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16298-5_1
12. Durán, F., Salaün, G.: Verifying timed BPMN processes using maude. In: Jacquet, J.-M., Massink, M. (eds.) COORDINATION 2017. LNCS, vol. 10319, pp. 219–236. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59746-1_12

13. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: Bubenko, J., Krogstie, J., Pastor, O., Pernici, B., Rolland, C., Sølvberg, A. (eds.) Seminal Contributions to Information Systems Engineering: 25 Years CAiSE, pp. 191–205. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36926-1_15

14. Hunsberger, L., Posenato, R., Combi, C.: A sound-and-complete propagation-based algorithm for checking the dynamic consistency of conditional simple temporal networks. In: 22nd International Symposium on Temporal Representation and Reasoning (TIME 2015), pp. 4–18 (2015). https://doi.org/10.1109/TIME.2015.26

15. Jurdziński, M., Trivedi, A.: Reachability-time games on timed automata. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 838–849. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73420-8_72

16. Kon, E., et al.: Platelet-rich plasma intra-articular injection versus hyaluronic acid viscosupplementation as treatments for cartilage pathology. Arthrosc. Relat. Surg. **27**(11), 1490–1501 (2011). https://doi.org/10.1016/j.arthro.2011.05.011

17. Kumar, A., Barton, R.: Controlled violation of temporal process constraints-models, algorithms and results. Inf. Syst. **64**, 410–424 (2017). https://doi.org/10.1016/j.is.2016.06.003

18. Kumar, A., Sabbella, S.R., Barton, R.R.: Managing controlled violation of temporal process constraints. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 280–296. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_20

19. Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. J. Softw.: Evol. Process **23**(4), 205–244 (2011)

20. Lanz, A., Reichert, M., Weber, B.: Process time patterns: a formal foundation. Inf. Syst. **57**, 38–68 (2016). https://doi.org/10.1016/j.is.2015.10.002

21. Morris, P.H., Muscettola, N., Vidal, T.: Dynamic control of plans with temporal uncertainty. In: International Joint Conference on Artificial Intelligence (IJCAI 2001), pp. 494–502 (2001)

22. Object Management Group: Business Process Model and Notation (BPMN) Version 2.0 (2011). http://www.omg.org/spec/BPMN/2.0. Accessed 07 June 2018

23. Pichler, H., Eder, J., Ciglic, M.: Modelling processes with time-dependent control structures. In: Mayr, H.C., Guizzardi, G., Ma, H., Pastor, O. (eds.) ER 2017. LNCS, vol. 10650, pp. 50–58. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69904-2_4

24. Posenato, R., Lanz, A., Combi, C., Reichert, M.: Managing time-awareness in modularized processes. Softw. Syst. Model. (2018). https://doi.org/10.1007/s10270-017-0643-4

25. Watahiki, K., Ishikawa, F., Hiraishi, K.: Formal verification of business processes with temporal and resource constraints. In: IEEE International Conference on Systems, Man, and Cybernetics, pp. 1173–1180 (2011). https://doi.org/10.1109/ICSMC.2011.6083857

26. Zavatteri, M.: Conditional simple temporal networks with uncertainty and decisions. In: 24th International Symposium on Temporal Representation and Reasoning (TIME 2017), pp. 23:1–23:17 (2017). https://doi.org/10.4230/LIPIcs.TIME.2017.23