A state-based context-aware declarative process model

Pnina Soffer and Tomer Yehezkel

University of Haifa, Carmel Mountain 31905, Haifa, Israel
spnina@is.haifa.ac.il, yehezkel.tomer@gmail.com

Abstract. Declarative process models support process flexibility, which importance has been widely recognized, particularly for organizations that face frequent changes and variable stimuli from their environment. However, current declarative approaches emphasize activities, thus the expressiveness of their constraints is limited. This expressiveness is not capable of addressing the process context (namely, environment effects) and its goal. The paper proposes a declarative model which addresses activities as well as states, external events, and goals. As such, it explicitly addresses the context of a process. The model is based on the Generic Process Model (GPM), extended by a notion of activity, which includes a state change aspect and an intentional aspect. The achievement of the intention of an activity may depend on events in the environment and is hence not certain. The paper provides a formalization of the model and some conditions for verification. These are illustrated by an example from the medical domain.

1 Introduction

The importance of flexibility in process aware information systems has been widely acknowledged in the past few years. Flexibility is the ability to make changes in adaptation to a need, while keeping the essence unchanged [10]. Considering business processes, flexibility is the ability to deal with both foreseen and unforeseen changes, by varying or adapting specific parts of the business process, while retaining the essence of the parts that are not or should not be impacted by the variations [12].

Flexibility is particularly important in organizations that face frequent changes and variable stimuli from their environment. For processes that operate in a relatively stable environment, when unpredictable situations are not frequent, flexibility is not essential, as responses to all predictable situations can be defined. However, in the present business environment, where changes occur frequently and organizations have to cope with a high range of diversity, full predictability is quite rare.

Facing this reality, approaches have been proposed for enabling flexibility in business processes, as reviewed and classified in [12]. These include mechanisms of late binding and modeling, where the actual realization of a specific action is only decided at runtime as implemented in YAWL [1], and changes that can be made at runtime to a running process instance or to all instances of the process, enabled in ADEPT [11].

One of the promising approaches is declarative process models (e.g., Declare [9]), which have received significant attention in recent years.

While “traditional” process models are imperative, explicitly specifying the execution order of activities through control flow constructs, a declarative process model is based on constraints, i.e., anything is possible as long as it is not explicitly forbidden. Constraint-based models, therefore, implicitly specify the execution procedure by means of constraints: any execution that does not violate constraints is possible.
Using such model, the user can decide how to respond to each situation that arises, executing an activity from among all the possibilities that are available in compliance to the specified constraints. While allowing a high degree of freedom, this approach is limited for the following reasons.
First, while the human decision about which action to take is made based on the state at that specific moment, the existing models do not emphasize states. Rather, the leading concept to be modeled and monitored in the model is an activity, and constraints can be specified on the execution of a single activity or on relationships between activity executions. The process state is monitored, mainly as a trace of the activities that have been executed up to a given moment. Constraints can also relate to values of data as conditions for activity execution. However, there is no fundamental view and monitoring of state for leading process execution and decision making.
Second, to respond to changes and events that occur in the environment, these need to be addressed in the model. Generally speaking, the model should be context aware, where context is the set of inputs a process instance receives from its environment. This is particularly important when bearing in mind that flexibility is required in the first place in processes that face frequent changes in the environment.
Finally, an effective selection of action by the human operator of the process should relate to the desired outcomes to be achieved, namely, to a goal. Currently, goals are usually not an integral part of process definitions.
This paper outlines semantics for a declarative process model to overcome the three discussed limitations. To enable the development of a consistent and complete model, we rely on the Generic Process Model (GPM) [16], which is an ontology-based theoretical framework for process analysis. GPM uses states as a leading element in process representation; it has been used for analyzing the context of processes [6], and it includes goals as basic building blocks in process specification.
Since GPM emphasizes states and abstracts from activities in a process model, in this paper it is amended to cater for activities as well.
In what follows, we start by a motivating example, demonstrating the limitations of Declare, as a representative activity-based declarative model. We then present the concepts required for our declarative model, first by informally deriving them from GPM, and then as formal definitions that set the basis for execution semantics. The use of our concepts for designing and validating processes is demonstrated through application to the running example. This is followed by discussion of related work, conclusions, and outlining of future research directions.

2 Motivating Example

This section presents a motivating example of a CT virtual cardiac catheterization process, which will be used throughout the paper as a running example. A Declare

model of the process is given in Fig. 1. The process starts with a pre CT evaluation of the patient (marked in the model as "init"). This evaluation may find the patient not fit for the scan, in which case the patient is released and the process ends. If the patient is fit and is not regularly on beta blockers, he will be administered beta blockers as preparation for the scan. Following this, either obesity (for overweight patients) or regular cardiac CT scan is performed once. The CT scan uses a low level of radiation, serving as a first indication of arteriosclerosis. If a positive indication (evidence of calcification) is obtained, the patient is released. If the first scanning does not discover a clear evidence of calcification, a second scanning is performed. Scanning can be performed up to twice (marked in the model as "0..2" for the scan activities), and if the second scan fails, the patient is released. In case the second scan is successful, its results are deciphered and interpreted. Deciphering can be successful or unsuccessful, but in any case the patient is released. At any point in the process, some acute health situation might be identified, in which case the patient is immediately sent to an emergency room (and the process ends). In the model this is represented as ER intervention, which has an exclusive choice relation with Release patient (both end the process under different circumstances). Another possibility is that the patient may feel bad during the process (due to allergy, claustrophobic reaction, irregular heart rate, etc.). In such cases the procedure may be paused for a while and resumed after a while, when the patient feels better. Additionally, at any point in time, the patient may be released so the process ends, but unsuccessfully.
The Declare model specifies the ordering of the process activities by a precedence relation, denoting that these activities normally follow one another, but not in all cases. The activities of ER intervention and Pause examination are not mandatory in the process, and are not related to other activities by any temporal constraint.

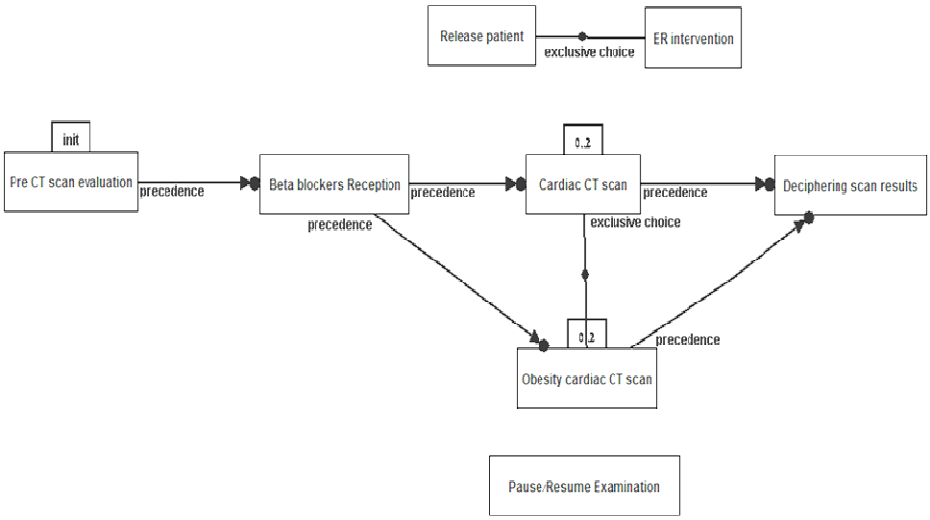

Fig. 1. The example process: a Declare model

The Declare representation supports the flexibility which is required for the process, catering for unforeseen situations and providing an immediate response based on

human decision making. It also allows defining data that serves as input or output to activities and using data as part of the activity relationship constraints. For example, the precedence between Pre CT evaluation and beta blockers reception is conditioned by a "fit" value of the data item Candidacy, assigned by the Pre CT evaluation activity.

However, we claim that this representation is not expressive enough, and it leaves parts of the flow logic of the process to human judgment, while this logic is clear and needs to be specified and enforced. Examples include: (1) a second scan is performed only if a clear evidence of calcification has not been obtained in the first scan; (2) Beta blockers reception is needed only for patients who do not use them regularly; (3) Pause examination and ER intervention are performed when the patient has some irregularity or when an acute problem is identified, respectively. These are constraints on the process flow, which cannot be expressed as relationships between activities or existence constraints on the activities. Furthermore, the model does not specify conditions under which the process terminates. Implicitly, the process cannot end before all the existence constraints on its activities are satisfied. However, in our example the only mandatory activity is Pre CT evaluation, while termination of the process is possible under defined conditions. It is possible to add a set of negation constraints, negating any activity after the activities of Release patient or ER intervention are performed. This, however, would result in a loaded model which is hard to follow.

Roughly speaking, we may conclude that Declare does not support constraints that relate to the context of the process and to its goal. Rather, these are assumed to be addressed by human judgment when the process is executed, enabled by the flexibility of the specification.

In the following sections we present an approach derived from theory, which enables a process specification that captures contextual constraints and process goals, while supporting flexibility. The theoretical basis provides for a complete set of constructs, capable of fully expressing the business logic of processes.

3 Ontological state-based view

The starting point of our discussion is the Generic Process Model [15][16], which is a process analysis framework, building on Bunge's ontology [5]. GPM emphasizes states, events, and goals, which, as shown above, are not well addressed in current declarative process models.

The focus of attention in GPM is the *domain* where the process takes place. The process domain is a *composite thing*, represented by a set of *state variables*, whose values at a moment in time denote the *state* of the domain. A state can be *unstable*, in which case it will transform according to the *transformation law* of the domain (*internal event*), or *stable*, namely, it will not change unless invoked by an event in the environment (*external event*). GPM views an enacted process as a set of state transitions in the process domain. Transitions result either from *transformations* within the domain (reflecting its transformation law), or from actions of the

environment on the domain. A process ends when the domain reaches a desired (*goal*) state, which is stable and where no more changes can occur due to domain dynamics. A process model is an abstract representation of the process, defined as follows.

Definition 1 *(GPM process model): A process model in a given domain is a tuple <I, G, L, E>, where*
I: the set of possible initial states – a subset of unstable states of the domain.
G: the goal set – a subset of the stable states reflecting stakeholders' objectives.
L: the transformation law defined on the domain – specifies possible state transitions as mappings between sets of states.
E: a set of relevant external events that can or need to occur during the process.

As noted, the focus of attention in GPM is the process domain. The domain sets the boundaries of what is fully controlled by the process and its operators, and what is not within control. This distinction enables us to define the context of a process [6] as the set of environmental effects on the process, which are twofold. First, the properties of the specific case handled by a process instance – these are assumed to exist at the initiation of the case, although not all their values are necessarily known at that point in time. Second, actions of the environment during process execution – these are manifested as external events. External events are events (state transitions) in the environment of the process domain, which affect the state of the domain through mutual state variables. Taking place outside the process domain, they are not controlled by it. The occurrence of an external event can be unanticipated, but even if we anticipate the occurrence, its exact time and its resulting state are usually not predictable. In particular, it is different for every process instance. Hence, the E and I elements in a process model represent contextual elements.
A second advantage of GPM is that it explicitly addresses the goal of a process, enabling the design of a process to achieve its goal, and assessing the validity of a process design against its defined goal. At runtime, achieving a goal state marks the termination of a process instance.
However, the transition law of GPM, which is a mapping between sets of states, is an abstract notion. Specifically, as indicated in Definition 1, GPM's process model abstracts from activities, which are how state changes are brought about. Hence, to make GPM an appropriate basis for declarative process models, the law needs to be decomposed into activities and constraints. To do so, a clear understanding of what an activity is needs to be developed.
Activities are the means for achieving internal events. Since internal events usually affect a subset of the domain state variables, namely, a sub-domain, and since different internal events can occur concurrently in independent sub-domains [14], we may address an activity as an internal event in a sub-domain. However, a sub-domain may change its state through a series of internal events in an almost continuous manner. What makes a specific trajectory be considered an activity is the intention that drives it. For example, consider the activity of Pre CT evaluation, which entails actions such as measuring the patient's blood pressure and heart rate, performing an electrocardiogram, and others. We consider all these actions as parts of one activity, distinguished by the one aim to be achieved. Intentions can be of achieving, maintaining, or avoiding a state [7].

We define an activity as an internal event in a sub-domain, intended to achieve a defined change in its state.
According to our model the state change brought by an activity is deterministic. However, the state variables whose values are changed might be mutual state variables of the process domain and its environment. In such cases, the environment is affected and its state might become unstable. This, in turn, causes transformations (events) in the environment, and these events might, again, affect the process domain. Thus, an activity that acts on the environment might lead to an external event in response. Since external events are not controlled by the process domain and their outcome is unpredictable, this might seem as if the outcome of the activity is unpredictable (especially if the reaction is immediate). Nevertheless, we specify only the controllable change within the process domain as part of the activity, and distinguish the uncontrollable change as an external event invoked by the activity and its effect on the process environment. For example, consider a basketball player throwing the ball to the basket. The activity ends once the ball is in the air, which is an unstable state of the environment. The movement of the ball in the air is not controlled by the player. The resulting event can be that the ball misses or hits the basket, and it is an external event, not completely predictable. The actual value resulting from the external event will be determined at runtime.
Note that in this example, the intention of the activity was to bring about a state where the ball is in the basket, but this can only be achieved by an external event, and with uncertainty.
Activities can hence be classified to two classes: (a) activities that affect only state variables which are intrinsic to the process domain. Such activities cause a fully predictable change that achieves the intention associated with the activity. (b) Activities that affect the state of the environment and invoke an external event. For these activities the specified (and predictable) change in the state does not necessarily correspond to the intended change. It may not even relate to the same state variables.
Note that activities of class (a), namely activities that operate on intrinsic domain state variables, may also entail changes in state variable values that depend on input given by the user at runtime. As an example, consider a process where the price of a product is determined. The activity of pricing might require the user to set a price based on his individual judgment of the appropriate profit margin. This will be manifested as user input at runtime, but is considered part of the activity since the value is controlled within the process domain.
Constraints:
The GPM specification can be represented by three types of constraints: (a) initiation constraints that set the possible initial set of states, (b) transformation constraints that specify the relationship state-activity, namely, states that are preconditions for activities, (c) Termination (goal) constraints that define the set of states where the process can terminate and be considered as having achieved its goal. In addition, we can define a fourth type of constraint – environment response constraints that place external events as response to activities that affect the environment. Note that the actual effect of these events on the domain is not known, nor is the exact time of their occurrence. We now discuss each of these four constraint types.
Initiation constraints – determine values of state variables to specify the conditions under which the process can begin (e.g., when a patient arrives at the clinic). In

particular, all the state variables that count the occurrences of activities are set to zero. Note that the initiation constraints do not determine the exact state on which a process instance begins. This exact state also includes values of contextual properties which characterize each specific case (e.g., the weight of a patient).

Transformation constraints – these include two kinds of constraints: enabling constraints and triggering constraints. Enabling constraints relate activities to the sets of states when they can be activated. Note that the state that follows the execution of an activity is directly calculated from the state that precedes it and the change it causes. Triggering constraints specify sets of states when an activity must be activated, so when a state in this set is reached the activity will immediately fire.

Termination constraints – determine sets of states where the process terminates. There might be two kinds of termination states. First, goal states, which are stable states the process is intended to achieve. Once a state in the goal set is reached, the process terminates. Second, exception states, which are stable states where the process terminates without achieving what it is intended to achieve. For example, the virtual cardiac catheterization process can terminate when it is found out the patient is not fit for scan or when an ER intervention is needed. These are exception states. Note that the process may include stable states which are not defined as termination states. If such a state is reached, the process waits for an external event to reactivate it. In the virtual cardiac catheterization process a state after the examination has been paused is stable, waiting for an external event when the patient feels better and has no irregularity to resume the process.

Environment response constraints – relate external events to activities that invoke them. Note that external events can also occur unexpectedly. Also note that in many cases the external event does not necessarily immediately follow the activity; there might be some time elapse between them. Hence, the relationship is of precedence.

Finally we note that it can be shown that the combination of initiation, termination, triggering, and enabling constraints is sufficient for expressing all the constraint types available in Declare. Our set of constraints provides these operations with respect to a broader scope, including context and goal. Hence, it provides a richer expressiveness. power.

4 Formalization

Following the above discussion, we now formalize the proposed constructs and execution semantics.

Definition 2 *(process model): Let D be a domain represented by its state variables vector* $X=(x_1,x_2,\ldots x_n)$*. Let* v_i *be the domain of values of state variable* x_i*,* $V=(v_1,v_2,\ldots v_n)$*. A process model M over D is a tuple (I, G, A, Const, E), where*

I: a set of states satisfying the initiation constraints

G: a set of states satisfying termination constraints; $G=Gg\cup Ge$*; Gg includes states defined as the goal of the process, Ge are states of exceptional termination.*

A: a set of activities

Const: a set of constraints

E: a set of external events.

In general, sets of states are specified by predicates over the state variable vector. Hence, given predicates C_I, C_{Gg}, and C_{Ge} that specify initiation, goal, and exceptional termination conditions respectively, we obtain:

$I=\{s \mid C_I(X)=TRUE\}$; $Gg=\{s \mid C_{Gg}(X)=TRUE\}$; $Ge=\{s \mid C_{Ge}(X)=TRUE\}$.

As discussed in the previous section, activities are intentional changes in the state of a sub-domain. Following this, the specification of an activity includes two elements: the change (delta) it brings about to the state of the sub-domain and the intended set of states to be achieved.

Definition 3 *(activitiy): Let $\delta(X)$ be a function, $\delta: V \rightarrow V$. Then $a \in A$: $(\delta(X), \gamma(X))$, where γ is a predicate denoting the set of states intended to be achieved by the activity.*

Note that δ usually implies a change in a subset of the domain state variables, which are the ones affected by the activity. In particular, a state variable counting the number of executions of the activity will be raised by 1. Also note that if $\gamma(X)$ includes negation operators, then the intention of the activity is to avoid a set of states. As well, if $\gamma(X)$ refers back to the set of states that precede the activity (except for the state variable that counts the executions of the activity), then the intention of the activity is to maintain an existing state.

The set of constraints includes the transformation and environment response constraints, since initiation and termination constraints are specified in *I* and *G*. As discussed in the previous section, transformation constraints include enabling constraints and triggering constraints.

Definition 4 *(enabling constraint): Let $a \in A$, θ_a a predicate, $En(a)=\{s \mid \theta_a(X)=TRUE\}$, then a can fire for every $s \in En(a)$.*

Definition 5 *(triggering constraint): Let $a \in A$, τ_a a predicate, $Tr(a)=\{s \mid \tau_a(X)=TRUE\}$, then a must fire for every $s \in Tr(a)$.*

In order to define the environment response constrains, we first need to define the external events element in the model. In a process model an external event is an occurrence we make no a-priori assumptions about (e.g., regarding its effect on the state of the domain). However, some external events which are expected to occur are expected to affect a subset of the domain state variables and assign them some value within its domain of possible values. The actual state that follows an external event will become known at runtime as input made by the user.

Definition 6 *(external event): An external event $e \in E$: $\{(x_i, v_i) \mid x_i \in X, v_i \in V\}$*

In words, an event is defined by a subset of the domain state variables which it affects, resulting in values within their domain of values.

Environment response constraints relate expected external events to the activity that invokes them.

Definition 7 *(environment response constraint): Let $a \in A$, $e \in E$. An environment response constraint Er:(a,e) denotes that e always occurs eventually after a.*

Since the occurrence of external events is not within the process control, environment response constraints cannot be enforced at runtime. Nevertheless, they are specified in

the model so they can be considered at process design time, and can be taken into account when planning ahead in runtime.

Based on the above definitions, we now provide a semi-formal description of the execution mechanism of a process. When a process is executed the state s at a moment in time is the values of x_i at that moment. When a process instance is created, the initial state is $s_I \in I$, satisfying the initiation constraints and including state variable values that represent contextual properties (initiated by user input). After initiation, the state at every moment is considered. For a given state s, the set of enabled activities is $A_{En}=\{a \mid s \in En(a)\}$; the set of triggered activities is $A_{Tr}=\{a \mid s \in Tr(a)\}$; an activity in A_{En} can be executed; an activity in A_{Tr} must be executed at that moment.

State changes can occur due to activity completion or to external events. Assume an activity a starts when the state is s. Then the state on completion of a is $\delta_a(s)$. The occurrence of event e requires the user to provide specific values for each state variable affected by the event; these values set the state that follows the event. Termination of the process is also determined based on the state, so if $s \in G$ then the process terminates.

Note that the intention component of the activity specification does not take part in the execution. However, it plays an important role at process design, as detailed in the next section. In addition, the intention is meaningful for planning ahead at runtime.

5 Specifying a process

This section demonstrates how the proposed semantics can be used for expressing the running example of the virtual cardiac catheterization process.

Table 1: State variables in the example process and their initial values

State variable	Values	Initial value	State variable	Values	Initial value
Compatibility	{Null, Fit, Not fit}	Null	Beta Blockers	{Given, Not given}	
Over-weight	{Yes, No}		Calcification	{Found, Not found}	Not found
Images	{Null, Successful, Unsuccessful}	Null	Deciphering results	{Null, Successful, Unsuccessful}	Null
Acute problem	{Undiscovered, Discovered}	Undiscovered	Irregularity	{Null, Appear, Disappear}	Null
Patient released	{Yes, No}	No	ER Intervention	{Yes, No}	No
Pre CT Evaluation	[0, ∞)	0	Beta blockers reception	[0, ∞)	0
Cardiac CT scan	[0, ∞)	0	Obesity CT scan	[0, ∞)	0
Deciphering scan results	[0, ∞)	0	Pause/resume examination	{0, 1}	0

We start by defining the state variables of the domain and their possible range of values (Table 1). Table 1 also provides the initial value of each state variable, defining the initial set of states of the process, *I*.

Note that initial values are set for a subset of the state variables, while state variables whose initial value is not specified stand for contextual properties. These need to be initialized to represent specific case properties. In our example process the relevant contextual properties are overweight of the patient and whether the patient is regularly on beta blockers. Also note a set of state variables that count the executions of each activity, as seen in their possible values – natural numbers from 0 to infinity.

The termination set is comprised of two sets of states, *Gg* of desired (goal) states and *Ge* of undesired termination states. Considering our example:

Gg={s | (Scan Deciphering = “successful”) $\wedge$ (Patient Released=”Yes”)}

Ge={s | ((Deciphering scan results $\neq$ “successful”) $\wedge$ (Patient Released=”Yes”)) $\vee$ (ER intervention=”Yes”)}

Ge stands for two possible cases of termination – when the patient is released without having reached successfully deciphered images (e.g., not found fit to scanning, or after calcification has been discovered), or when ER intervention is needed.

The activities of the process are specified in Table 2 in terms of the function δ, relating to specific state variables, and the predicate γ. The table also specifies for every activity *a* the related transformation constraints *En(a)* and *Tr(a)*.

To illustrate the specification of activities and their related transformation constraints, let us consider the activity Obesity CT scan, whose δ relates to the execution counter of the activity, raising it by 1. Recall, this activity can be performed up to twice. Ideally, after two executions a state will be reached where γ is achieved, namely *(Calcification ="Not Found")* $\wedge$ *(Images = "Successful")*. The enabling set of this activity is when *(Compatibility="Fit")* $\wedge$ *(Beta blockers = "Given")* $\wedge$ *(Over-weight = "Yes")* $\wedge$ *(Calcification = "Not Found")* $\wedge$ *(Paused/Resume Examination = 0)* $\wedge$ *(Obesity CT scan < 2)*, denoting that (a) the activity can start after beta blockers are given (either in the process or in its context) and compatibility is evaluated and found fit (this condition is needed in case beta blockers are given contextually), (b) the activity is executed only for patients with over-weight, (c) the activity can only be performed twice, and it is not repeated if calcification is found, and (d) the activity cannot start when the examination is paused.

As another example, consider the activity Pause/resume examination, whose role is to pause the examination when the patient has irregularities, and to resume it when the irregularity disappears. The activity can be triggered when irregularity appears if the examination is not already paused *((Irregularity = Appear)* $\wedge$ *(Paused/Resume Examination = 0))*, in which case the activity stops the examination (see *(Paused/Resume Examination* $\rightarrow$ *1) if (Paused/Resume Examination = 0)* in the δ column). Alternatively, the activity is triggered when the examination is already paused and the irregularity disappears, in which case the activity resumes the examination.

Note that there are activities such as Beta blockers reception, where for a state preceding the activity $s \in En(a) \cup Tr(a)$, the change achieved with certainty $\delta_a(s)$ satisfies the intention γ_a. These are activities that achieve their intention with certainty, not depending on external events. For other activities (e.g., Pre CT Evaluation), an external event is expected in response to the activity, for a state satisfying γ_a to be achieved. As previously discussed, in these cases the intention of

the activity may not be achieved, and it depends on the state variable values set by the external event.

Table 2: Activities and corresponding transformation constraints

Activity	**δ**	**γ**	**Transformation constraints**
Pre CT Evaluation	Pre CT Evaluation→ Pre CT Evaluation + 1	Candidacy= “fit”	En: (Candidacy=“null”)∧ (Paused/Resume Examination=0)
Beta blockers Reception	(Beta blockers Reception→Beta blockers Reception +1) ∧ (Beta blockers = "Given")	Beta blockers = "Given"	En:(Compatibility="Fit") ∧ (Beta blockers≠"Given")∧ (Paused/resume Examination = 0)
Cardiac CT scan	Cardiac CT scan) → Cardiac CT scan +1	(Calcification ="Not Found") ∧ (Images = "Successful")	En: (Compatibility="Fit") ∧ (Beta blockers = "Given") ∧ (Over-weight = "No") ∧ (Calcification = "Not Found") ∧ (Paused/Resume Examination = 0) ∧ (Cardiac CT scan < 2)
Obesity CT scan	Obesity CT scan) → Obesity CT scan +1	(Calcification ="Not Found") ∧ (Images = "Successful")	En: (Compatibility="Fit") ∧ (Beta blockers = "Given") ∧ (Over-weight = "Yes") ∧ (Calcification = "Not Found") ∧ (Paused/Resume Examination = 0) ∧ (Obesity CT scan < 2)
Deciphering scan results	Deciphering scan results→ Deciphering scan results + 1	Deciphering results= “Successful”	En: Images = "Successful"
ER Intervention	ER intervention = ”Yes”	ER intervention = ”Yes”	En: Patient released = “No” Tr: Acute problem= "Discovered"
Release Patient	Patient Released = “Yes”	Patient Released = “Yes”	En: (ER intervention=”No”)
Pause / resume examination	Paused/Resume Examination → 1) if (Paused/Resume Examination = 0); (Paused/Resume Examination → 0) if (Paused/Resume Examination = 1)	(Paused/Resume Examination=1∧ Irregularity= Appear) ∨ (Paused/Resume Examination=0∧ Irregularity= disappear)	Tr: ((Irregularity = Appear) ∧ (Paused/Resume Examination = 0)) ∨ ((Irregularity = Disappear) ∧ (Paused/Resume Examination = 1))

To complete the process specification, we define the set of external events *E*, which, together with the uninitiated variables in Table 1, form the context of the process. Table 3 includes external events and their associated environment response constraints (column “Response to” in the table). Some of the external events are expected in response to specific activities, while some can occur unexpectedly, in which case the “Response to” column is blank.

To illustrate the content of the table, consider the event Calcification discovery. This event is expected in response to a CT scan (either obesity or regular). It may change the value of the state variable Calcification from Not Found to Found (see Table 1).

Table 3: External events in the example process

External event	Affected state variables	Response to
Candidate compatibility	Candidacy	Pre CT scan evaluation
Calcification discovery	Calcification	Cardiac CT scan/ Obesity cardiac CT scan
Image generation	Images	Cardiac CT scan/ Obesity cardiac CT scan
Deciphering outcome	Deciphering results	Deciphering scan results
Acute health problem	Acute problem	
Irregularity appearance	Irregularity	

Having specified the process, we now present four conditions which are necessary for the specification to be valid, namely, for the process to achieve its goal.

Condition 1 ***(concurrency):*** *For every a, a'∈A, if* $Tr(a) \cap TR(a') \neq \varnothing$ *then* δ_a *and* $\delta_{a'}$ *do not affect the same state variables.*

This condition is intended to ensure that two activities that are triggered by the same state (namely, must be performed concurrently), do not change the values of shared state variables. For a discussion of this condition, see [14]. Our process includes two activities with triggering conditions: Pause/resume examination and ER intervention. Their triggering constraints are not overlapping, hence Condition 1 is not breeched.

Condition 2 ***(sequence of intended states):*** *There exists at least one sequence of states* $(s_1, s_2, \ldots s_n)$ *such that* $s_1 \in I$, $s_n \in Gg$, *and let* $s_i \in \{s|\gamma_{ai}=TRUE\}$ *then for* $i=2\ldots n$ $s_{i-1} \in En(a_i) \cup Tr(a_i)$.

This condition requires the existence of at least one sequence of states leading from an initial state to the goal of the process. Note that the sequence is established when the enabling or triggering set of each activity is in the intended set of the previous one. This means that the activity is enabled either immediately and certainly after an activity whose intention is achieved by its δ, or after an external event responding to the previous activity has achieved its intention.

In our example it can be noticed that for, e.g., the sequence of activities Pre CT scan evaluation, Beta blockers reception, Cardiac CT scan, Deciphering scan results, and Release patient, the enabling set of each activity satisfies the intention of the previous one. As well, the first activity (Pre CT scan evaluation) is enabled at I and the last one can lead to a state in Gg.

Also note that there might be other sequences which do not lead to the goal set. These, however, should end on a termination state in Ge and not on any other state. In other words, continuation of the process should be enabled for any state which is not in G. To ensure this continuation, we require the following two conditions.

Condition 3 ***(process continuation - activities):*** *Let* $a \in A$ *such that* γ_a *is not achieved by* δ_a. *Then* $\exists$ *an external event* $e \in E$ *and an environment response constraint* $Er:(a,e)$.

For example, δ of Deciphering scan results increases the execution counter of this activity by 1, while its intention is to reach a state where Deciphering results are Successful. This can be achieved by the external event Deciphering outcome, which is related to the activity by an environment response constraint.

Condition 4 ***(process continuation - events):*** *For every external event e: {(x_i, v_i)}, for every value v_i that state variable x_i can assume, the resulting state s satisfies (1) $s \in G$ or (2) $\exists$ activity $a \in A$ such that $s \in En(a) \cup Tr(a)$.*
This condition requires that every external event either leads to a state in the termination set or to a state where at least one activity is enabled / triggered. For example, the event Acute health problem leads to a state where *Acute problem = "Discovered"*. This state triggers the activity of ER intervention.

6 Related work

Substantial research efforts have been invested in declarative process models in recent years. Most notably, Declare [9], a modeling and execution platform, which also allows changing the model at runtime and performing some verification. Methodological issues that concern declarative process models have also been investigated. Examples include life-cycle support [18], user assistance [12], and usability evaluation [17]. Life-cycle support [18] is said to ensure better understandability and maintainability of declarative processes over the process life-cycle, based on the ideas of Test Driven Development [4] and Automated Acceptance. User assistance includes recommendations which are generated based on similar past process executions by considering specific business objectives [13]. An initial usability evaluation using the Alaska simulator, has indicated that humans are capable of coping with flexibility and can effectively plan in an agile manner [17].
Declarative model segments have also been used for managing imperative models. Ly et. al. [8] developed a framework for integrating constraints into adaptive process management systems in order to ensure semantic correctness of running processes at any time. Awad et al. [3] suggested a technique to accomplish the verification of process models against imposed compliance rules by using BPMN-Q queries.
All these approaches basically employ an activity-based view, with Linear temporal logic (LTL)-based constraints. These constraints are capable of defining rules on the existence of activities and on dependencies among them.
Goals are usually not specified or addressed, although some goal consideration is included in the Alaska simulator [17]. The tool uses a journey as metaphor for business processes and determines typical goals such as the overall business value of the journey, i.e., minimization of cost, cycle time or the optimization of quality or customer satisfaction. Goals of this kind are not addressed in this paper (in GPM terminology they are called soft-goals [15]). Rather, we address "hard" goals which mark the termination of the process, while soft-goals mainly affect planning and will be addressed as future research.
A different and early approach is presented by [1], who defined business process patterns based on a state-oriented approach that includes a state space, a goal, and valid movements in the state space. Constraints are not made formally and explicitly, but roughly in the form of valid movements. The movements refer only to the changes in the constructed state space and abstract from activities.
In summary, as compared to the existing declarative process modeling approaches, the approach presented in this paper has an extended expressiveness, enabling all

LTL-based constraints and adding state information. Furthermore, it supports contextual constraints, which are not possible in existing approaches.

7 Conclusion

Declarative process models support flexibility in process aware information systems. However, current declarative models are mainly activity-based, relying on Linear Temporal Logic for the constraints they entail. As a result, the business logic related to the context of the process and to its goal is basically applied by human and not supported by the model.

The model proposed in this paper constrains the execution of activities based on state, which reflects activity execution as well as case properties and results of events in the environment of the process. As such, it is context aware and suitable for highly diverse and frequently changing environments, where process flexibility is particularly important. Furthermore, an explicit goal specification can guide execution towards this goal and serve for validating the process at design time.

A theoretical contribution of the paper is the activity definition, which makes a clear distinction between the certain change brought about by it and the intended change, which may or may not depend on environment response invoked by the activity. The intentional aspect of an activity is shown to be of importance for designing the process and for planning ahead for reaching the goal. Yet, it can be ignored by an execution engine for simplicity, as it has no role in the actual execution mechanism.

The paper demonstrates the specification of an example process using the proposed model. This specification, however, is not graphical. An appropriate graphical representation to increase the usability of the model by humans is still needed. We intend to consider the adaptation of the graphical notation used by Declare to the additional expressiveness required by our model as future work. The paper also provides four conditions a process specification needs to meet to be valid. As noted, these are necessary conditions for the validity of the process, but not necessarily sufficient. Full validation and verification of a process specification is also planned as future research, as well as the utilization of AI planning algorithms to support goal achievement from a given state.

References

[1] van der Aalst WMP and ter Hofstede AHM, YAWL: Yet Another Workflow Language. *Information Systems*, 30(4), pp. 245–275, 2005.

[2] Andersson B., Bider I., Johannesson P. and Perjons E., Towards a Formal Definition of Goal-Oriented Business Process Patterns, *Business Process Management Journal* (11:6), pp. 650-662, 2005.

[3] Awad, A., Decker, G., Weske, M., Efficient compliance checking using BPMN-Q and temporal logic, In: Dumas et. al. (eds), *BPM '08*, LNCS 5240, pp. 326-341, Springer-Verlag, Berlin, 2008.

[4] Beck K., Test Driven Development: By Example. Addison-Wesley, 2002.

[5] Bunge, M. 1977. *Treatise on Basic Philosophy: Vol. 3, Ontology I: The Furniture of the World*, Boston: Reidel.
[6] Ghattas J, Soffer P, Peleg M. A formal model for process context learning. In: *Proc. BPI, BPM Workshops 2009*, LNBIP 43, pp. 140-157, 2009.
[7] Lamsweerde A., Goal-Oriented Requirements Engineering: A Guided Tour, *5th Int'l Symp. on RE*, pp.249-261, IEEE CS Press, 2001.
[8] Ly, L.T., Rinderle, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. *Data and Knowledge Engineering* 64, pp. 3-23, 2008
[9] Pesic M, Schonenberg MH, Sidorova N, van der Aalst WMP, Constraint-based workflow models: Change made easy. In: Curbera et. al. (eds), *Proc. of OTM*, LNCS 4803, pp 77–94. Springer-Verlag, Berlin, 2007
[10] Regev G, Bider I., and Wegmann A. Defining business process flexibility with the help of invariants. *Software Process Improvement and Practice*, 12, pp. 65–79, 2007.
[11] Reichert M., Rinderle S., and Dadam P., Adept workflow management system. In van der Aalst et. al (eds), *BPM 2003*, LNCS 2678, pp. 370–379. Springer-Verlag Berlin, 2003.
[12] Schonenberg H., Mans R., Russell N., Mulyar N. and van der Aalst WMP, Process Flexibility: A Survey of Contemporary Approaches, In Dietz et. al. (eds), *CIAO! And EOMAS 2008*, LNBIP 10, pp. 16-30, Springer-Verlag, Berlin, 2008.
[13] Schonenberg H, Weber B, van Dongen BF, van der Aalst WMP, Supporting flexible processes through recommendations based on history. In: Dumas et. al. (eds), *BPM 2008*, LNCS 5240, pp 51–66, Springer-Verlag, Berlin, 2008.
[14] Soffer, P., Kaner, M., and Wand, Y. Assigning Ontology-Based Semantics to Workflow nets", *Journal of Database Management*, (21:3) pp. 1-35, 2010.
[15] Soffer, P., and Wand, Y. On the Notion of Soft Goals in Business Process Modeling, *Business Process Management Journal* (11:6) pp. 663-679, 2005.
[16] Soffer, P., and Wand, Y. ,Goal-driven multi-process analysis, *Journal of the Association of Information Systems* (8:3), pp. 175-203, 2007.
[17] Weber, B., Pinggera, J., Zugal, S., Wild, W.: Handling events during business process execution: An empirical test.
[18] Weber, B., Reijers, H., Zugal, S., Wild, W., The declarative approach to business process execution: An empirical test, In: van Eck et. al. (eds), *CAiSE 2009*, LNCS 5565, pp. 470-485, Springer-Verlag Berlin Heidelberg, 2009.
[19] Zugal S., Pinggera J., and Weber B.,Toward Enhanced Life-Cycle Support for Declarative Processes, *University of Innsbruck* ,2010.