

Business process languages: an ontology-based perspective

Greta ADAMO ^{a,c} and Stefano BORGIO ^b and Chiara DI FRANCESCOMARINO ^a and Chiara GHIDINI ^a and Nicola GUARINO ^b and Emilio M. SANFILIPPO ^b

^a*FBK-IRST, Via Sommarive 18, 38050 Trento, Italy*

^b*ISTC-CNR Laboratory for Applied Ontology, Trento, Italy*

^c*University of Genova, DIBRIS, via Dodecaneso 35, 16146 Italy*

Abstract. Business process modelling (BPM) notations describe processes using a graphical representation of process-relevant entities and their interplay. Despite the wide literature on the comparison between different modelling languages, the BPM community still lacks an ontological characterisation of process constructs. Purpose of this paper is to start filling this gap by providing a first ontological analysis of the main business process entities. The analysis and the resulting characterisation aim at illustrating the different perspectives that BPM languages implicitly take on business processes, as well as guiding the modellers in making an appropriate choice when selecting among different notations.

1. Introduction

Business process modelling (BPM) notations describe processes using a graphical representation of process-relevant entities and their interplay. If we focus on typical business-to-consumer (B2C) scenarios, examples of languages include well-known imperative languages such as the Business Process Model and Notation (BPMN), the Unified Modeling Language Activity Diagram (UML-AD) and the Event-driven Process Chain (EPC) as well as declarative notations such as the Case Management Model and Notation (CMMN) and DECLARE.¹ Despite the wide literature on process execution semantics and on the comparison between the graphical constructs of different languages [25, 14, 12, 16], the BPM community still lacks a robust ontological characterisation of the entities involved in process models.² While some initial efforts have been done towards this direction (see, e.g., [20]), they focus on the analysis of behavioral aspects of process models, thus neglecting other central modelling constructs such as those denoting process participants (data objects, actors and so on). As a result, process participants are exposed to a paradox: on the one hand, they are explicitly referred to within process diagrams; on the

¹Imperative paradigms aim at producing models that describe all allowed flows: every flow that is not specified in the model is implicitly disallowed. Declarative process modelling notations instead allow the production of flexible models obtained by describing constraints on the allowed flows: all flows are allowed provided that they do not violate the specified constraints.

²We will interchangeably use the notions of ‘process model’ and ‘process diagram’.

other hand, they are emblematically neglected when explaining or illustrating the very notion of process in the BPM community.

The purpose of the paper is to remove the above paradox, offering an ontological analysis of the various kinds of process participants, with a specific focus on B2C scenarios that will hopefully contribute to the ontological foundations of BPM. The paper is structured as follows. In Section 2, we provide an illustration of different constructs used by imperative and declarative modelling notations; then we identify in Section 3 the constructs referring to process participants, and, after analysing the definition of process in Section 4, we discuss the ontological properties of process participants in Section 5, providing in this way an initial comparison of (some of) the modelling constructs used in different modelling notations (Section 6). This represents a first step toward the illustration of how an ontological analysis enlarged to process participants can support the interpretation of business process diagrams, the comparison between modelling notations, the illustration of the different perspectives that BPM languages implicitly take on business processes, as well as guiding the modellers in making an appropriate choice when selecting among different notations.

2. Background

In this section we illustrate the graphical elements of the BPM languages taken into account throughout the paper.³ As a starting point, we select five amongst the most popular languages that follow the imperative (BPMN, UML-AD, EPC) and declarative (CMMN and DECLARE) paradigms. To support the presentation, we make use of process diagrams illustrating the simple scenario of a customer buying a flight ticket from a travel agency. For the sake of clarity, we “annotate” the diagrams with speech balloons to explicitly indicate the graphical constructs.

BPMN. It is a standard language, proposed by the Object Management Group (OMG), to design business processes.⁴ BPMN defines a Business Process Diagram (BPD) which includes a set of graphical constructs divided in: (i) flow objects, (ii) data, (iii) connecting objects, and (iv) swimlanes. Flow objects define the behavior of a business process, as the one reported in Figure 1. They divide into *events*, *activities* and *gateways*. Events represent things that happen during a process; they divide into *start*, *intermediate* and *end* events. An activity is a generic term that is used for work to be performed. It can be either atomic (*task*) or compound (*sub-process*). A gateway determines the forking, merging or joining of paths. BPMN 2.0 allows for the explicit modelling of data by means of constructs denoting *data objects*, *data inputs*, *data output* and *data stores*. Flow objects are inter-linked through *connecting objects* which are not further discussed here. *Swimlanes* are used to specify who is responsible for the execution of a certain process.⁵ Looking at Figure 1, for example, two swimlanes specify that ‘Customer’ and ‘Travel agency’ will carry out the depicted processes.

³Note that our analysis focuses on the graphical elements used to draw models and leave out further notions that may be included in the language specification.

⁴<http://www.omg.org/spec/BPMN/2.0/>

⁵Because of lack of space we do not introduce the distinction between *pools* and *lanes* here.

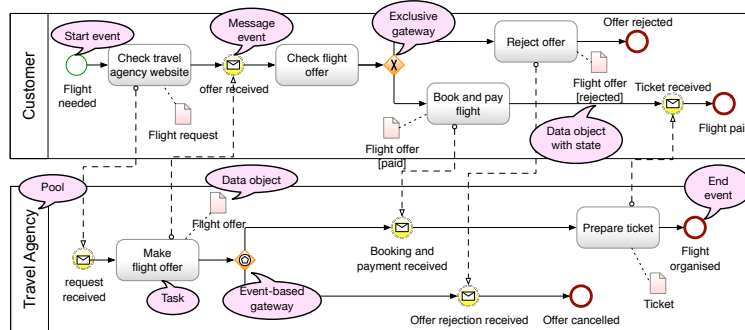


Figure 1. A Business Process Diagram in the BPMN language.

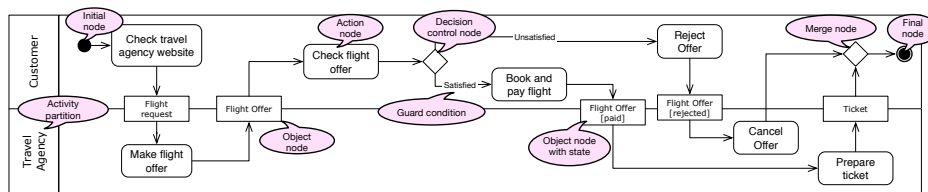


Figure 2. A Business Process Diagram in the UML AD language.

UML-AD. It is one of the diagram families of the OMG standardized UML language⁶, whose purpose is to describe the control and data flow as a sequence of activity nodes connected by activity edges (see example in Figure 2).

The nodes responsible of describing the control flow are the *action nodes* and the *control nodes*. While the former represent atomic steps within an activity, the latter allow for controlling the execution flow by means of the AND, OR or XOR logical operations. Additional control flow nodes are used to depict the initial and final nodes of process models. *Object nodes* and *object flows* are the main UML-ADs constructs describing the data flow. The former represent objects at a given point of the flow and, as such, they can also have an associated *state*. The latter are instead used for connecting object nodes to actions. *Activity partitions* are a mechanism for grouping activity nodes that have common characteristics. They are mainly used to define organizational units. Finally, the notation allows for specifying activity pre- and post-conditions, for instance, by annotating activity edges with guards.

EPC. It is a modelling language developed in the early 1990s as part of the Architecture of Integrated Information Systems (ARIS) framework [22].

Three types of nodes are responsible for describing the control flow: *function*, *event* and *logical operators* (see Figure 3). Function nodes represent atomic activities and can be considered as the “active” part of a control flow; event nodes stand for the states in which a process happens to be and can be therefore considered as the “passive” part of the control flow. Functions and events alternate, capturing the intuition that states lead to activities, while activities generate states. Finally, the XOR, AND and OR logical operators allow for controlling the execution flow.

⁶<http://www.omg.org/spec/UML/>

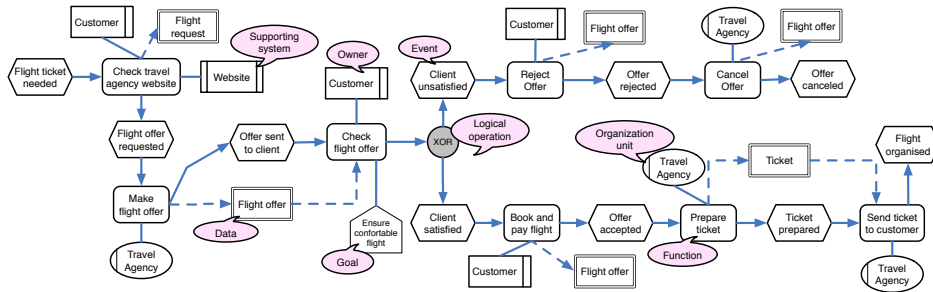


Figure 3. A Business Process Diagram in the EPC language.

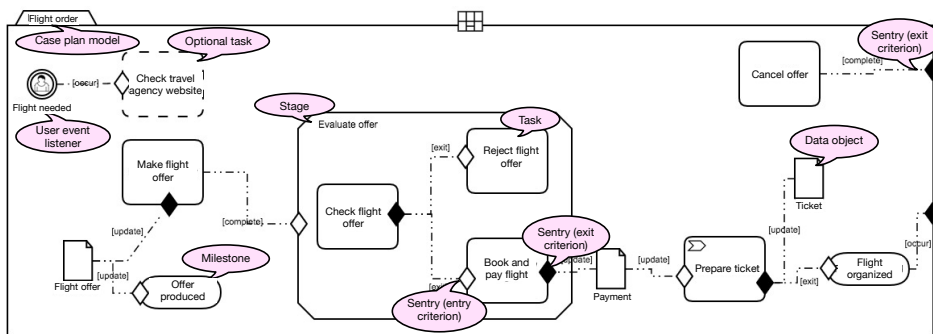


Figure 4. A Business Process Diagram in the CMMN language.

Functions within the control flow can be connected to objects belonging to the other views of an ARIS model, namely the organizational, data, function and product service views. While the exact number of objects differs across different versions of the language,⁷ the core modeling constructs usually denote: (i) input and output *data, material, services or resource objects* required or produced by a function; (ii) *owners* who are responsible for a specific function; (iii) *organization units* responsible for a specific function (e.g., a department); and (iv) *supporting systems* upon which a function acts (e.g., a database). Some versions include *goals* that can be connected to specific functions.

CMMN. It is a OMG standard for the declarative representation of process models.⁸ Its main modelling construct is the *case*, which is described by a case diagram (see Figure 4). Differently from the previous languages, CMMN follows a declarative approach. Thus, rather than describing all the allowed flows of a process from the start to the end, it models cases as composed of process segments (called *stages*) and *tasks*.

A case plan model contains: (possibly discretionary) *tasks, stages, milestones, event listeners, connectors, and sentries*. A task is a unit of work. Stages are plan fragments which can be composite or atomic. A milestone represents an accomplishment which occurs during the process of a case. Events represent something that can happen to a plan construct (e.g., a task cancelled) or in general (timer and user event listener). Connectors

⁷The analysis and diagrams contained in this paper refer to the description provided in [23].

⁸<http://www.omg.org/spec/CMMN/1.1/>





TEMPLATE	NOTATION	DESCRIPTION
init(A)		A must occur as a first activity
not coexistence(A,B)		If A occurs, B must not occur and viceversa
precedence(A,B)		B can occur only if A has occurred before
response(A,B)		if A occurs, then B occurs after A

Table 1. Graphical notation of some DECLARE templates.

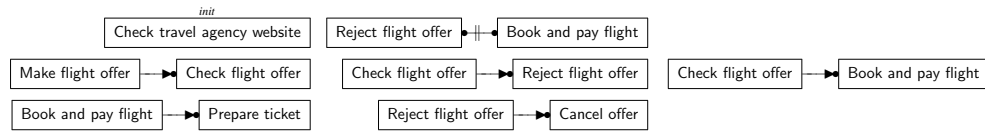


Figure 5. A DECLARE process diagram.

are used to link different plan items. Finally, sentries represent the entry / exit criteria for path items and can direct the control flow mimicking the AND and OR logical operators.

Declare. It is one of the most popular declarative languages for modelling business processes [18]. It grounds on the finite-trace semantics of LTL and aims at capturing variable cases by means of the so-called *patterns*. These are particular LTL formulae that have been singled out for process modelling taking inspiration from [8]. Table 1 reports the graphical notation and a brief description of the patterns used across the paper; Figure 5 provides a DECLARE version of the flight ticket example using the patterns in the table. As we can see from the figure, DECLARE focuses only on the temporal relations between (atomic) activities and does not provide any support for data or actors.⁹ Combinations of patterns, such as *precedence* and *non-coexistence*, can be used to direct the control flow using the AND, OR, and XOR logical operators.

3. A brief comparison of business process language constructs

We present now a short categorization and comparative summary of the main modelling constructs of the five languages. Modelling constructs are grouped into the three basic categories of process modelling languages, namely the *behavioural* (BEV) category related to the control flow, the *data* (DT) category related to the data flow, and the *organizational* (ORG) category related to process actors. Since the behavioural category is the most articulated, we further describe it in terms of *Functional* (executable pro-active actions), *Event* (what happens), *Flow* (how actions are connected and routed), and *State* (of the world) categories. The result of this grouping is summarized in Table 2.

First, we can observe that all the imperative languages, namely BPMN, EPC, and UML-AD, provide distinctive constructs to indicate the start and the end of a process.

⁹Some proposal to extend DECLARE in order to incorporate non-atomic activities and data centric patterns are presented in [6] and [2], respectively. They are nonetheless more focused on the logical properties of the specification rather than on the definition of modelling constructs suitable for business analysts, and are therefore not considered in this paper.

	BPMN	UML-AD	EPC	CMMN	DECLARE	
BEV	Func	Task	Action node	Function	Task	
		Subprocess	Activity	Process path	Stage	
	Event	Start/End	Start/End node		Timer	
		Intermediate	Accept event action	–	User Event Listener	–
		Send/receive	Send signal action			
	Flow	Gateway	Control node	Logical operators	Connector	Connector
Sequence Flow		Control Flow	Control Flow	Sentry	Pattern	
Message Flow		Object Flow	Info Flow			
State	Guard on gateway	Guard on control node	Event	Sentry	–	
		Pre- Post-condition on activity	Start/End event	Milestone		
DT	Data input data output data store	Object node	(I/O) data object	Case file item	–	
ORG	Pool, Lane	Activity Partition	Organization Activity Owner	–		

Table 2. A comparison among modelling languages

CMMN specifies only exiting conditions while DECLARE allows (but does not force) to specify only initial activities. Not surprisingly, all five languages have graphical symbols for atomic activities. Instead, subprocesses and generic groups of activities are foreseen in all languages but EPCs and DECLARE. Other common constructs are routing nodes, connectors and data objects. CMMN (DECLARE) does not have explicit construct for routing nodes; nonetheless a combination of sentries and connectors, (DECLARE patterns) can be used to route the control flow mimicking the logical operators. The level of details of connectors can vary. Besides the one used to denote connections of the control flow, common to all languages, BPMN, EPC, and UML-AD provide symbols to denote the connections between actors (data) and activities, or the messages exchanged between different activities. Also, the level of detail of data objects can vary; e.g., EPC is particularly rich in defining a taxonomy of data objects. Alternative (OR, XOR) routing nodes can incorporate guards, i.e., conditions that specify which branch to follow, in all languages but EPC, where this role can be taken by states. Actors and organizational constructs are present in imperative languages, although exploiting different notations.¹⁰

A distinction that is present in BPMN (to some extent also in CMMN) is between active tasks, explicitly performed by the actor specified in a corresponding swimlane, and passive events that occur independently from the actor itself. Other distinctive aspects are (i) the explicit presence of pre- (activation) and post-conditions on activities, which is one of the characteristic features of CMMN and is also foreseen in UML-AD; (ii) the explicit presence of a state, which is a characteristic feature of EPCs where states and functions (tasks) have to interleave, and is also present in CMMN in the form of milestones.

4. On the definition of business process

Davenport [5] defines a business process as “a structured, measured set of activities designed to produce a specific output for a particular customer or market. [...] A process is thus a specific ordering of work activities across time and space, with a beginning and an end, and clearly defined inputs and outputs”. Similar definitions are provided by

¹⁰Note that CMMN allows to associate organizational entities to cases during the run-time phase.

Hamer and Champy [11], and Johansson et al. [13]. The first states that a business process is “a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer”; the latter says that it is “a set of linked activities that take an input and transform it to create an output. Ideally, the transformation that occurs in the process should add value to the input”. The most modern and popular definition in the BPM literature is likely the one provided by Weske [26], who defines a business process as “a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations.”

From these definitions, it is rather spread the idea of a business process as a set of **activities** that, together, contribute to achieve a certain **goal** or, more in general, contribute to transform an **input** into a desired **output**. Besides the emphasis on these core aspects, the literature underlines the importance of additional characteristics such as the **value** brought about from the realisation of a certain goal, as well as the **organisational boundaries** in which a process is embedded.

Despite this general agreement, the BPM literature does not provide in depth explanations for what a process is or what its components (e.g., activities) are. Let us consider two illustrative examples. First, it remains unclear whether processes are defined at the type or at the instance (execution) level. A process execution happens in time and has a specific duration. Differently, process types are descriptions and do not unfold in time.¹¹ Examples of process types are depicted in process diagrams like the ones in the previous sections that describe how the various activities are connected to produce a goal, which is in turn a description of a desired state (e.g., that a certain ticket is purchased). The carrying out of these activities in real-time (possibly monitored by so-called event logs) provides the execution of the process type. As an example, process models may be only descriptive (and not prescriptive) and process executions non compliant with the process model are often considered executions of that process by people from the BPM community. Indeed all the process mining activities tend to define processes at the execution level more than at the process diagram level.

Second, the BPM literature does not provide in depth explanations of the intended semantics of the various modelling constructs, e.g., what an activity is, what its participants are, and how the latter relate to the former. An emblematic example is the lack of characterisation of the relations that occur between the activities in a process. While a business process is invariably understood as an ordered collection of activities, it remains unclear, e.g., whether such activities are only temporally or also causally related, or whether the effects of activities need to be explicitly represented.

5. An ontological analysis of some business process constructs

In this section, we provide an analysis of some of the modelling constructs we encountered in the previous sections, with particular emphasis on (the lack of) an ontological characterisation of their properties. We rely on previous works (e.g., [21,20]) for the characterisation of process-like entities (activities, tasks, and events), whereas we shall

¹¹Concerning the type-execution dichotomy, see the distinction between `Activity` and `ActivityOccurrence` in the *Process Specification Language* (PSL) [9], respectively.

dig into the analysis of process participants. The results of the analysis are used in Section 6 to provide an initial ontologically grounded characterisation of the business process modelling languages summarised in Table 2.

Activities In the BPM literature, activities are understood as (atomic or compound) *actions*, consisting of intentional transformations from some initial state (the input) to some other state (the output). The participants to such actions are the entities that take part in these transformations. From the ontological point of view, actions are (specific kinds of) *events*, while their participants are *objects* [4].

A first aspect that needs to be considered concerns the relation between activities, and more in general the way the activities contribute to the achievement of the goal. In general terms, the precedence relation between activities can be a temporal, causal or dependence relation, perhaps depending on the context or scenario to be modelled. For instance, assume that in a slight variation of the example of Section 2, the travel agency splits the activity ‘Make flight offer’ in two subsequent steps ‘Send flight offer to customer’ and ‘Archive offer’ which, for purely organisational reasons, must be executed in this order. This would be a pure temporal relation between the activities in this specific setting and could easily be modified in a revision of the process model. Instead the activity of ‘Paying for a flight’ may be a strong precondition for the ‘Preparation of the ticket’, and so it should necessarily occur before the latter. Nonetheless these relations would be denoted by means of the same connector symbol.

Another aspect that needs to be considered is the (explicit or implicit) representation of the world’s states affected by the designed process. While the definitions of process in Section 4 describe business processes as a way to move from an input (state) to an output (state), they do not specify whether it is desirable to explicitly represent the world’s states affected by the designed process. Nonetheless, the description of intermediate states of affairs can be often found in business process models. Think for instance to functions in EPC diagrams, the status of data objects in BPM and UML-AD, and sentries and milestones in CMMN. Thus, a question that one may ask is whether the (explicit or implicit) representation of the world’s states is necessary to fully characterise a process (model) and what are the characteristics of this representation.

Participants From a general perspective, participants can be *physical* or *non-physical* depending on their properties. In fact, the very same activity may involve several types of objects as participants: physical objects (e.g., the knife used to cut a piece of bread); information objects (e.g., personal data involved in submitting a request); agents and/or organizations playing certain roles (e.g., an administrative employee receiving a form).

A physical participant, whenever it exists¹², is located in the physical space. Differently, non-physical participants lack physical locations, although they are present in time. A person is an example of physical participant, whereas the content of a person’s ID, which is an information object (see below), is a non-physical participant. Information objects are rather common in business processes and are represented by means of data objects modelling constructs. In applied ontology, only a few systems [24,15,3,17] have attempted a formal treatment of information. These ontologies agree in distinguishing between information objects and their physical carriers like paper sheets or pdf files; also, the same information object may be encoded in multiple carriers while retaining its

¹²We use the expressions ‘to exist’ and ‘to be present in time’ as synonyms.

identity. For example, John's and Mary's copies of the *Divine Comedy* are two different carriers of the same information object. Information objects and their physical carriers have an important role in business processes. For example, in the scenario of buying a flight ticket, we may consider the flight offer an information object whose physical carrier is not that relevant. Instead, the physical carrier (e.g., a pdf file) is of fundamental importance to exchange the ticket. Again, a question that one may ask is whether this distinction needs to be represented in a process (model) and what are the characteristics of this representation.

Another crucial distinction in BPM is the one between *agentive* and *non-agentive* participants. For the purposes of our work, we consider a process participant as agentive depending on whether it has sensors, actuators and the capability to act on itself or on the environment. For instance, a lathe machine is an agent when, e.g., it has sensors by which it acquires data from the objects to be manufactured and acts upon them by elaborating the data through some software. Additionally, we consider a second notion of agentive participant characterising an agent that can be ascribed with intentions, including beliefs or desires. Non-agentive participants that undergo a change during an action are usually called *patients* of the action. It is easy to see that a process such as the one in Section 2 contains both agentive (e.g., the customer paying for the flight) and non-agentive participants (e.g., the offer whose status changed from created to rejected).

Apart from the classification of participants, we need to spend some words on their roles. From an ontological perspective, the latter are properties that objects only contingently satisfy within certain contexts, e.g., processes (e.g., *to be a resource* during a drilling process) or organisations (*to be professor* at MIT). In this sense, an object can loose or acquire a role while remaining the same entity. We assume that roles can be ascribed to any type of participant, including information objects. The ability to constrain the way an object playing a role participates in a process may be also of interest to the BPM field, not only at the execution but also at the type level. Let us focus, for example, on organisational/business roles, which in BPMN are usually described by means of pools, such as the pools 'customer' and 'travel agent' in Figure 1. While it seems plausible to assume that the customer does not change during the entire duration of the process (otherwise this would be another process instance), the same constraint would not apply to the 'travel agent'. In fact, any employee of the travel agency playing the 'travel agent' role would perfectly fit the specification of this process. Enriching BPM languages with the ability to distinguish between these cases may be useful to constrain and reason on the identity of process instances.

The second column of Table 3 summarises the main characteristics of business processes and of business process constructs described here and in Section 4. In the next section we provide some insights on whether the modelling notations described in Section 2 enable to represent these characteristics, and how. The results are summarised in the right hand side of Table 3.

6. Discussion

In this section, we discuss the ontological aspects of business processes presented in Section 5 in the light of the modelling constructs presented in Section 2, with the help of the flight purchase example. The results are summarised in Table 3

	CHARACTERISTIC	BPMN	UML-AD	EPC	CMMN	DECLARE
PROCESS	Set of activities	Yes	Yes	Yes	Yes	Yes
	Clear Input/Output	Yes	Yes	Yes	Somehow	Somehow
	Goal/Value	No	No	Somehow	Somehow	No
	Organizational boundaries	Yes	Yes	Yes	No	No
ACTIV.	Different types of relations between activities	No	No	No	No	Somehow
	State of the word	Somehow	Somehow	Yes	Somehow	No
PART.	Agentive vs non agentive	Somehow	No	Somehow	No	No
	Information vs carrier	No	No	No	No	No
	Object vs role	Somehow	Somehow	Somehow	No	No

Table 3. A comparison among modelling languages

By looking at the diagrams, we observe that all the notations enable to represent structured / coordinated sets of activities. This is not surprising: the specification of the control flow is indeed the top priority of a process model. The situation changes as soon as we move to the clear specification of input and output states. Here we observe that all the notations but DECLARE enable / require an explicit initial and final state. Not surprisingly, the imperative modelling languages (BPMN, UML-AD and EPC) strictly require explicit start and end symbols / states. These languages, in fact, model business processes in a prescriptive manner specifying all the allowed flows from a start (the input) to an end (the output) state. Despite its declarative nature, CMMN also facilitates the modellers to specify input and output states by means of input and output sentries and by explicitly asking for exit criteria. Instead, DECLARE is, in our opinion, the weakest language in terms of driving the modeller to explicitly represent input and output states. In fact, it provides a (optional) pattern for the initial activity, and it does not foresee any pattern for the exit / last activity, thus lacking also a way to express - even if implicitly - the goal, or desired state, of a business process. Moving to the specification of the business goal or added value that the business process realises, none of the modelling languages force the modeller to make them explicit. The only language that explicitly contains a ‘goal’ construct is (one of the variants of) EPC. Thus, we can say that most BPM languages leave implicit in the modeller’s (and the reader’s) mind the goal the activities contribute to realise. Finally, organization boundaries can be easily described in notations such as BPMN, UML-AD and EPCs, using notions such as pools/lanes, activity partitions, and organization/activity owner, respectively, while they are absent in CMMN and DECLARE.

Moving to the relation between activities, we can easily notice that almost all the languages only enable a connection between activities without specifying the nature of such connection. A notable exception is DECLARE, whose main focus is indeed the representation of (temporal) relations between activities. While it would be incorrect to say that DECLARE patterns have the aim of specifying the kind of relation existing between different activities, it is also true that some (temporal) patterns may be better suited to model causal vs. temporal vs. dependent relations. As an example, let us consider the ‘response’ and ‘precedence’ patterns in Table 1. While the precedence pattern may be suited to express that an activity happens before another, and therefore can be considered as a pure temporal constraint, the ‘response’ pattern conveys the meaning that *B* is a consequence of *A* being true (happening). Thus we may say that, from an ontological perspective, DECLARE somehow guides the modellers to think about the type of relation existing between activities, besides the simple sequencing.

A key difference among the modelling notations we took into account concerns the representation of the (state of the) world in response to a process execution. Figure 3 emphasises this as one of the focuses of EPCs. UML-AD and CMMN lie in the middle by exploiting data objects and sentries for describing how the world is changed because of the process execution. BPMN, instead, only provides (optional) constructs for representing the state of data objects. Finally, DECLARE does not offer any construct at all for representing the status of the world. From an ontological perspective, we would say that, differently from DECLARE and BPMN, EPC drives the modeller to explicitly represent the world's states affected by the designed process, while UML-AD and CMMN guide the modeller to implicitly represent the world's states through data objects and sentries.

The participants relevant in the flight purchase example are the customer, the travel agency and various information objects. The process thus includes different types of participants, material and immaterial ones. A first observation we have to make is that DECLARE does not offer any support to the modelling of entities that participate to the activities. It is therefore ignored in the remaining of the discussion. By looking at the diagrams, we observe that no explicit distinction is made between agentive and non-agentive participants. Nonetheless, the specification of activity owners in EPCs seems to suggest that they have the ability to act. Also, pools in BPMN are understood as participants in collaboration, and therefore exhibiting the capability to collaborate. In case of UML-AD, the activity partitions may be used for grouping activities with different purposes, i.e. not only according to the activity performer; however, when used for this purpose, also the UML-AD notation tends to suggest the ability of the performer to act. CMMN, instead, does not seem to distinguish between these types of participants.

In Figures 1–3, *Check travel agency website* results in the *Flight request* which is sent to the agency. On the basis of our analysis, one has to distinguish between the request-information-object and the request-support(s); to some extent, the former is more relevant than the latter, since it represents the customer's information to book the flight. However, one cannot avoid referencing the support. Hence, what the customer sends to the agency is a (copy of a) physical object displaying an information object. The distinction between information object and support is not addressed by the languages we considered; rather, it is blurred in the notion of data object.

No actors appear in CMMN and neither BPMN nor UML-AD specify whether 'customer' explicitly refers to a single individual (e.g., John) or to an organisation. In both cases, it reasonably stands for the *role* of a participant, who desires to book a flight ticket. This consideration reveals, besides the lack of declarative language graphical constructs for specifying actors, the underspecification of both BPMN and UML-AD with respect to our analysis, since pools and activity partitions can be used to refer to different types of participants, but also to their roles.¹³ Differently, the distinction between single actors and organisations can be explicitly conveyed in EPC, although the difference between participants and their roles is blurred.

To conclude, the analysis of process entities needs to be extended to identify the different modelling approaches in the languages at hand. Once we recognise the ontological assumptions undergoing the different languages, including the lack of characterisation of several properties, we can better understand how to correctly use the language to convey

¹³Although some support is given in the meta-models of the different languages, what we aim at emphasising here is the lack of support provided by the graphical notations of the different languages.

a well characterised meaning. This latter topic however deserves more attention and is left for future work.

7. Related and Future Work

Focusing on *ontology-based* BPM, which is the context of our paper, disparate ontologies have been proposed to semantically enrich process models. Among these, some ontologies axiomatise the properties that graphical constructs satisfy according to modelling notations (see, e.g., [19]). In a more general setting, an upper-level ontology for business processes is proposed in [7]. In these works, however, the authors do not attempt an ontological clarification of the modelling notations at stake. Some initial works towards the analysis of BPMN based on foundational ontologies are presented in [10,20]. These however focus only on constructs like activities and events, while leaving aside the analysis of participants, which is the focus of the presented work. Related work focused on the provision of semantic foundations for role-related concepts in the context of enterprise modelling is presented in [1].

In the future we plan to extend our preliminary analysis in order to deepen the investigation of the ontological commitments of modelling notations by further inspecting the different perspectives that they implicitly take on business processes and their participants, as well as by providing modellers with guidelines to make an appropriate choice when selecting among different notations. We also aim at extending our analysis including further languages that encompass the B2C view, such as ArchiMate, Petri Nets or the Integrated DEFINITION Methods (IDEF) language.

Acknowledgments This research has been partially carried out within the Euregio IPN12 KAOS, which is funded by the “European Region Tyrol-South Tyrol-Trentino” (EGTC) under the first call for basic research projects.

References

- [1] J. P. A. Almeida, G. Guizzardi, and P. S. Santos, Jr. Applying and extending a semantic foundation for role-related concepts in enterprise modelling. *Enterp. Inf. Syst.*, 3(3):253–277, Aug. 2009.
- [2] A. Artale, M. Montali, S. Tritini, and W. van der Aalst. Object-centric behavioral constraints: Integrating data and declarative process modelling. In *Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017*, CEUR Workshop Proceedings, 2017.
- [3] C. Bekiari, M. Doerr, P. Le Bœuf, and P. Riva. FRBR object-oriented definition and mapping from FRBRER, FRAD and FRASAD (version 2.4). *International Working Group on FRBR and CIDOC CRM Harmonisation*, 2015.
- [4] S. Borgo and C. Masolo. Foundational choices in DOLCE. In S. Staab and R. Studer, editors, *Handbook on Ontologies*. Springer Science & Business Media, 2013.
- [5] T. Davenport. *Process Innovation: Reengineering work through information technology*. Harvard Business School Press, Boston, 1993.
- [6] R. De Masellis, C. D. Francescomarino, C. Ghidini, and F. M. Maggi. Declarative process models: Different ways to be hierarchical. In *Proc. of the 14th Int. Conf. on Service-Oriented Computing (IC-SOC2016)*, volume 9936 of *LNCS*, pages 104–119. Springer, 2016.
- [7] A. De Nicola, M. Lezoche, and M. Missikoff. An ontological approach to business process modeling. In *3th Indian International Conference on Artificial Intelligence 2007*, pages ISBN–978, 2007.
- [8] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *Proc. of the 1999 International Conf. on Software Engineering (ICSE)*. ACM Press, 1999.

- [9] M. Grüninger. Using the PSL ontology. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 423–443. Springer-Verlag Berlin Heidelberg, 2009.
- [10] G. Guizzardi and G. Wagner. Can BPMN be used for making simulation models? In *Workshop on Enterprise and Organizational Modeling and Simulation*, pages 100–115. Springer, 2011.
- [11] M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. Harper Business, 1993.
- [12] F. Heidari, P. Loucopoulos, F. M. T. Brazier, and J. Barjis. A meta-meta-model for seven business process modeling languages. In *IEEE 15th Conference on Business Informatics, CBI 2013*, pages 216–221. IEEE Computer Society, 2013.
- [13] H. J. Johansson, P. McHugh, A. J. Pendlebury, and W. A. Wheeler. *Business Process Reengineering: Breakpoint Strategies for Market Dominance*. John Wiley & Sons, 1993.
- [14] B. List and B. Korherr. An evaluation of conceptual business process modelling languages. In *Proc. of the 2006 ACM Symposium on Applied Computing, SAC '06*, pages 1532–1539. ACM, 2006.
- [15] C. Masolo, L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. Social roles and their descriptions. In *Principles of Knowledge Representation and Reasoning*, pages 267–277. AAAI Press, 2004.
- [16] H. Mili, G. Tremblay, G. B. Jaoude, E. Lefebvre, L. Elabed, and G. E. Boussaidi. Business process modeling languages: Sorting through the alphabet soup. *ACM Comput. Surv.*, 43(1):4:1–4:56, 2010.
- [17] R. Mizoguchi. Yamato: yet another more advanced top-level ontology. In *Proceedings of the Sixth Australasian Ontology Workshop*, pages 1–16, 2010.
- [18] M. Pesic, H. Schonenberg, and W. van der Aalst. DECLARE: Full Support for Loosely-Structured Processes. In *EDOC*, pages 287–300, 2007.
- [19] M. Rospocher, C. Ghidini, and L. Serafini. An ontology for the business process modelling notation. In *FOIS*, pages 133–146, 2014.
- [20] E. M. Sanfilippo, S. Borgo, and C. Masolo. Events and activities: Is there an ontology behind bpmn? In *FOIS*, pages 147–156, 2014.
- [21] P. S. Santos Jr., J. P. Almeida, and G. Guizzardi. An ontology-based semantic foundation for aris eps. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 124–130. ACM, 2010.
- [22] A. Scheer. *ARIS - vom Geschäftsprozess zum Anwendungssystem*. Springer, Berlin [u.a.], 4., durchges. Aufl. edition, 2002.
- [23] A.-W. Scheer, O. Thomas, and O. Adam. *Process-Aware Information Systems: Bridging People and Software Through Process Technology*, chapter Process Modeling Using Event-Driven Process Chains, pages 119–146. Wiley, October 2005.
- [24] B. Smith and W. Ceusters. Aboutness: Towards foundations for the information artifact ontology. In *Proceedings of the International Conference on Biomedical Ontology (ICBO) 2015*, 2015.
- [25] E. Söderström, B. Andersson, P. Johannesson, E. Perjons, and B. Wangler. *Towards a Framework for Comparing Process Modelling Languages*, pages 600–611. Springer Berlin, 2002.
- [26] M. Weske. *Business Process Management. Concepts, Languages, Architectures*. Springer, 2012.