Can BPMN Be Used for Making Simulation Models?

Giancarlo Guizzardi[1] and Gerd Wagner[2],

[1] Computer Science Department, Federal University of Espírito Santo (UFES),
Av. Fernando Ferrari, s/n29060-970 Vitória, Espírito Santo, BRAZIL
gguizzardi@inf.ufes.br

[2] Institute of Informatics, Brandenburg University of Technology,
P.O.Box 101344, 03013 Cottbus, GERMANY
G.Wagner@tu-cottbus.de

Abstract. We investigate the question if BPMN can be used for simulation modeling by evaluating it against ABDESO, a foundational ontology for agent-based discrete event simulation. Since a simulation model is intended to capture a real-world system, BPMN, if used as the modeling language for making the simulation model, should have a "real-world semantics" based on a foundational ontology for agent-based discrete event simulation.

Keywords: BPMN, agent-based simulation, foundational ontology.

1 Introduction

Business process management (BPM) includes both business process modeling and business process simulation. Often, the animation of a business process model is considered to be a *business process simulation*. However, such a simulation is not really *business simulation* in the full sense, since this is rather understood as a simulation of the *business system*, and not of just one of its processes. A business system is a social system (an organization) having one or more actors that are involved in zero or more business processes at any time. Since actors are essential for business systems, it is natural to use agent-based simulation for business simulation.

The Business Process Modeling Notation [1] is today's de-facto business process modeling standard. It is considered to be understandable by all kinds of users, from the business analysts who create the initial drafts of the business processes, to the technical developers responsible for writing the software applications that will perform those processes, and finally, to the business people who will manage and monitor them. BPMN can be used both for making intuitive, non-executable business process models and for making executable models, such as needed for business process simulation.

In this paper, we investigate the question if a business system simulation model can be obtained by a set of executable BPMN models, which exhaustively describe its process types together with the actors participating in them. We approach this question by evaluating BPMN against ABDESO, a foundational ontology for agent-based discrete event simulation that has been proposed in [11]. Since a simulation model is intended to capture a real-world system, BPMN, if used as the modeling language for making the simulation model, should have a "real-world semantics" based on a foundational ontology for agent-based discrete event simulation.

In recent years, there has been a growing interest in the application of foundational ontologies (also known as upper level, or top-level ontologies) for providing real-world semantics for conceptual modeling languages, and theoretically sound foundations and methodological guidelines for evaluating and improving the models made using these languages. While the value of an ontologically well-founded conceptual modeling language is widely acknowledged in the areas of information system and software system engineering, as indicated by the great number of recent publications in this area, the issue of investigating the ontological foundations of simulation languages did not yet receive much attention in the scientific literature.

While there are several research papers on the ontological foundations of organization and business process modeling (see section 2), there has been no attempt yet to demonstrate the value of an ontologically well-founded modeling language for ABS. The main benefit obtained from establishing the onto-logical foundations of the core concepts of agent-based modeling languages is a clarification of their real world semantics. An ontological semantics of a simulation modeling language leads to a higher overall quality of the simulation models expressed in that language with respect to comprehensibility, maintain-ability, interoperability and evolvability.

In a series of publications [6], [7], [8], [9] we have reported about our project for developing a foundational ontology called "UFO" (for Unified Foundational Ontology) by employing theories from Formal Ontology, Cognitive Psychology, Linguistics, Philosophy of Language and Philosophical Logics. The core of UFO has been established through the development of an ontology of endurants by the first author in [5] This foundational ontology has been successfully applied in the analysis of several important conceptual modeling constructs such as Roles, Types, Part-Whole Relations, Attributes, and Datatypes, among others.

While the ontological foundations of basic DES have been analyzed in [10], using the discrete event system ontology DESO based on the UFO layers A and B (about objects and events), the ontological foundations of agent-based DES have been analyzed in [11], using the agent-based discrete event system ontology ABDESO based on the UFO layer C about agents.

The remaining of this article is organized as follows. In section 2, we discuss some related work. Section 3 contains a summary of UFO, and of ABDESO. Then, in section 4, we summarize the essentials of BPMN, and finally, in section 5, we use ABDESO for evaluating the suitability of BPMN as a business simulation modeling language.

3 Related Work

In [3], using the Web ontology language OWL, an ontology defining an agent-based simulation framework is presented and possibilities for using OWL's automated reasoning capabilities are discussed.

In [12], it is proposed to use ontologies (in the sense of conceptual domain models) for making the scientists' conceptual models more coherent with the simulation program code. This amounts to making an explicit conceptual model (using UML and/or OWL) before starting to code a simulation. However, although the paper refers to philosophical work on ontologies, foundational ontologies are not considered.

There is a large body of work, in which foundational ontologies are used for evaluating business process modeling languages, see, e.g [4]. As an example of more recent work on investigating the ontological foundations of multi-agent systems, see [2], in which the ontological modeling of organizations is discussed.

So, while there have been several proposals about how to use ontology engineering technologies, such as OWL, in agent-based simulation engineering, and there is a large body of work on using foundational ontologies for evaluating business process modeling languages, including BPMN, we were not able to find any work on the evaluation of BPMN as a business simulation language.

4 UFO and ABDESO

Since the development of UFO is an ongoing project, we use a simplified version of it, called *Essential Unified Foundational Ontology (eUFO)*, which restricts both the breadth and the depth of UFO, and simplifies its philosophical terminology, harmonizing it with informatics terminology as much as possible.

In this section, for making the present paper self-contained, we briefly summarize the base layer of eUFO, called eUFO-0, as well as its layer eUFO-*A* about substance individuals and trope individuals, and its layer eUFO-*B* about events, using UML class diagrams. These layers of eUFO have been more extensively discussed in [10].

eUFO-0 defines a number of basic ontological categories, as depicted in Figure 1 below, making a fundamental distinction between ***individuals***, which are things that exist in time and space in "the real world" and have a unique identity, and ***universal***, which are feature-based classifiers that classify, at any moment in time, a set of individuals with common features.

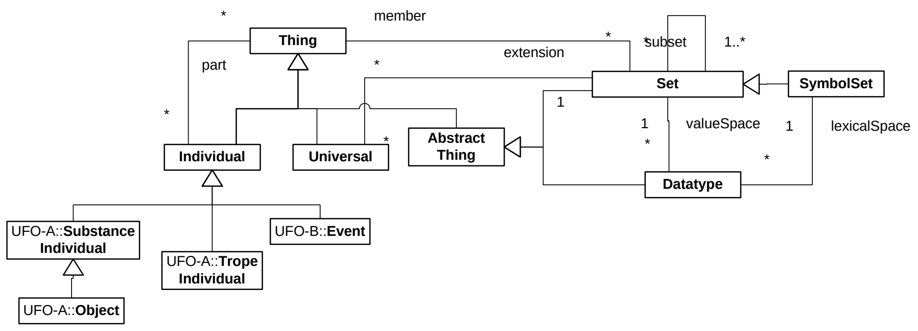

Fig. 1. eUFO-0 − the base layer of eUFO

We distinguish between three kinds of individuals: substance individuals, trope individuals and events. As opposed to substance individuals, trope individuals can only exist in other individuals, i.e., they are existentially dependent on other individuals. The distinction between substance individuals and events can be understood in terms of their relationship to time. Substance individuals are wholly present whenever they are present, i.e., they are in tim*e*. Events *happen in time*, they may have temporal parts.

4.1 Substance Individuals, Attributions, Relationships and References

Examples of substance individuals are: the person with name "Gerd Wagner", the moon, or an amount of sand. Examples of events are: today's rise of the sun, my confirmation of an e-commerce purchase order through clicking the OK button, or the Second World War. Examples of trope individuals are: the redness of John's T-shirt, Giancarlo's employment with UFES, or my daughter's belief in God.

The ontology of substance individuals and trope individuals forms the UFO layer *A*, which is depicted in Figure 2, while the ontology of events forms the UFO layer *B*, which is depicted in Figure 3.

There are two kinds of trope individuals: (a) *Intrinsic* trope individuals can be ***qualities*** such as an individualized color or a temperature, and ***modes*** such as a skill, a belief, or an intention; (b) *Relational* trope individuals or ***relators***: a medical treatment, a purchase order, or a social commitment. While qualities and modes depend on one single individual (their *bearer*), in which they *inhere*, relators depend on two or more individuals (their *relata*), which they *mediate*.

We distinguish between the color of a particular apple (as a *quality* of the apple) and the color *data value* that we associate with this quality in an ***attribution*** (with the help of an *attribute*). This data value is a member of the *value space* of the *data type* of the attribute. As an example, consider the attribute *hairColor*, which is applicable to persons, and associated to a datatype with a value space consisting of color names. Then, the triple <john, hairColor, grey> represents an attribution that makes the sentence "The hair color of John is grey" true.

While a *formal relationship*, such as [Brandenburg is part of Germany], holds directly, for a ***material relationship***, such as [Paul is being treated in the medical unit M], to exist, something else, which *mediates* the involved individuals (Paul and M), must exist. Such a mediating individual with the power of connecting individuals is called a *relator*. For example, a medical treatment connects a patient with a medical unit; an enrollment connects a student with an educational institution; a covalent bond connects two atoms. In general, relators are founded on events.

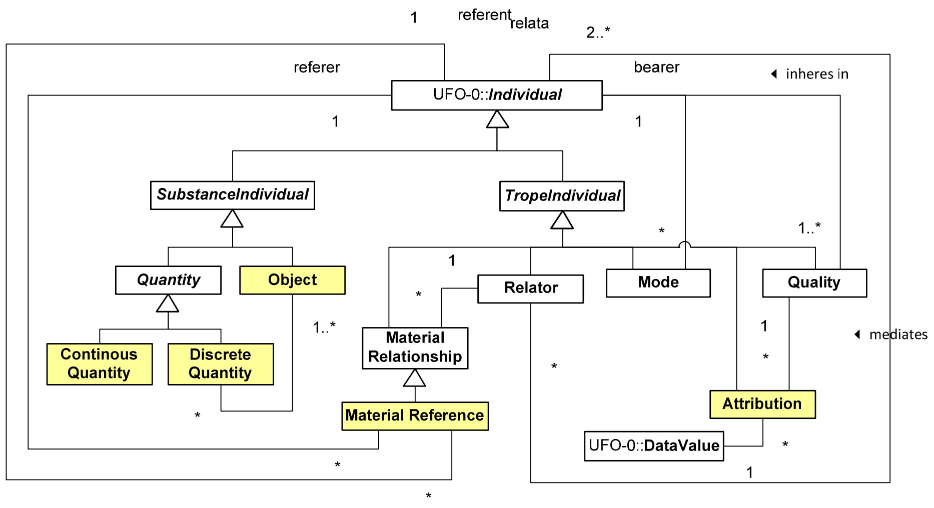

Fig. 2. eUFO-A – an ontology of substance individuals.

In a correspondence theory of truth (such as Tarski's semantics of predicate logic), attributions, references and relationships are considered as "truth makers" ("facts") that make corresponding sentences true.

4.2 Events

Events are individuals that may be composed of temporal parts. They *happen in time* in the sense that they may extend in time accumulating temporal parts. An event cannot exhibit change in time like a substance individual.

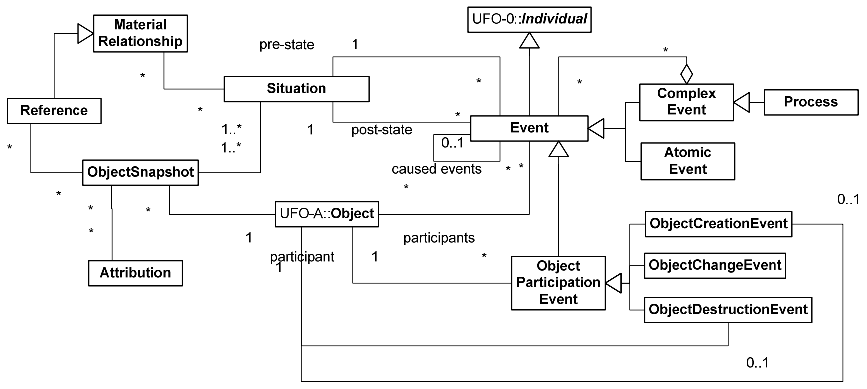

Fig. 3. eUFO-B – an ontology of events.

Figure 3 depicts the core fragment of the eUFO-B ontology of events, which can be atomic or complex, depending on their mereological structure. Events existentially depend on their *participants* in order to exist. For instance, in the event of Caesar being stabbed by Brutus, we have the participation of Caesar himself, of Brutus and of the knife. Each of these participations is itself an event (an *object participation event*), which existentially depends on a single object. Special cases of object participation events are *object creation*, *object change* and *object destruction* events.

Events may change the real world by changing the state of affairs from a *pre-state* situation to a *post-state* situation. Each situation is determined by a set of associated *object snapshots* and a set of associated material relationships holding between the involved objects, where an object snapshot is a set of attributions and references of a particular object.

4.3 Universals

Universals *classify* individuals, which are said to be their *instances*. The set of all instances of a universal is called its *extension*. We consider seven kinds of universals: *event types*, *object types*, *quality universals*, *attributes*, *relator universals*, *reference properties* and *material relationship types*. There are other kinds of universals, but these seven are the most relevant for conceptual modeling.

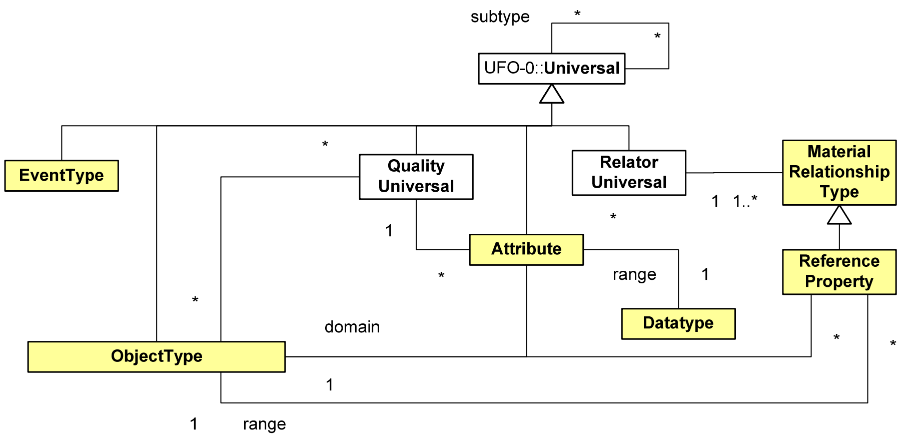

Fig. 4. eUFO-U – an ontology of universals (types).

While the notions of attribute, relationship type and reference property are well-known in computer science in the area of information and database modeling, their ontological foundation in connection with quality universals and relator universals is not well-known.

Different Kinds of Object Types

We distinguish between the different kinds of object types shown in Figure 5.

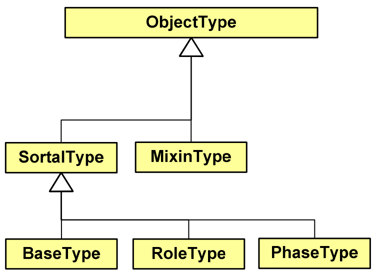

Fig. 5. The different kinds of object types of eUFO-U.

While all object types carry a principle of application, only *sortal types* carry a principle of identity for their instances. A principle of application allows to judge whether an individual is an instance of that object type. In contrast, a principle of identity allows to judge whether two individuals are the same. Non-sortal types, such as RedThing, are called *mixin types*.

Within the category of sortal types, we make a further distinction based on the formal notions of rigidity and anti-rigidity: A sortal type U is *rigid* if for every instance x of U, x is necessarily (in the modal sense) an instance of U. In other words, if x instantiates U in a given world w, then x must instantiate U in every possible world w'. In contrast, a sortal type U is *anti-rigid* if for every instance x of U, x is possibly (in the modal sense) not an instance of U. In other words, if x instantiates U in a given world w, then there must be a possible world w' in which x does not instantiate U. We call a rigid sortal type a *base type*.

For any anti-rigid object type A, there is a unique ultimate base type B, such that: (i) A is a subtype of B; (ii) B supplies the unique principle of identity obeyed by the instances of A. There is a specialization condition SC such that x is an instance of A iff x is an instance of B that satisfies SC. A further clarification on the different types of specialization conditions allows us to distinguish between two different kinds of anti-rigid object types: *roles* and *phase types*. Phase types constitute possible stages in the history of an individual.

For any role (e.g. `Student`) there is an underlying binary *material relationship type* or *reference property* (e.g. `students`) such that the extension of the role (e.g. the set of current students of an educational institution) is the *range* of that reference property.

Quality Universals and Attributes

A ***quality universal*** classifies individual qualities of the same type. A quality universal can be associated with one or more datatypes, such that any particular quality corresponds to a specific data value from the value space of the datatype. The association between qualities from some quality universal and the corresponding data values from an associated datatype is provided by an ***attribute***, which is a universal that classifies *attributions*. A quality universal can be captured by one or more corresponding attributes, each of them based on a different datatype.

E.g., the quality universal "hair color" could be captured by an attribute with the range of RGB byte triples or by an attribute with the range of natural language color names. Consequently, we may have more than one attribution for a particular quality, one for each associated attribute.

Relator Universals, Material Relationship Types and Reference Properties

A relator universal classifies individual relators of the same type. The ***material relationship type*** *R* induced by a relator universal ***R*** classifies all material relationships induced by relators from ***R***. Since each material relationship corresponds to a tuple, *R* also has a *tuple extension* (i.e. a *relation* in the sense of set theory). A material relationship type is a universal that classifies material relationships, which are 'truth makers' for material relationship statements.

A ***reference property*** represents a binary material relationship type, corresponding to a relator universal whose instances mediate exactly two individuals. Its tuple extension is a subset of the Cartesian product of the extensions of the two involved types. The first type is called the ***domain***, and the second one the ***range*** of the reference property.

4.4 DESO - A Foundational Ontology for Basic DES

We summarize DESO, a foundational Discrete Event System Ontology on the basis of eUFO, proposed in [10].

The Run-Time Ontology DESO-I

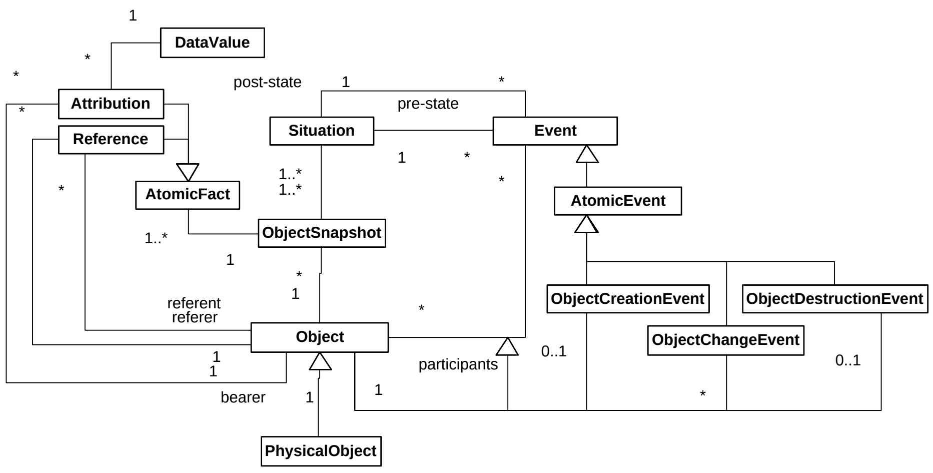

Fig. 6. The categories of individuals, as defined by DESO-I.

In DESO-I, for simplicity, we assume that there are only binary relationships, which are represented by ***references*** specifying an object as the value of a reference property. An ***atomic fact*** is either a reference or an attribution, as depicted in Figure 6.

Notice that in conceptual modeling, and in simulation modeling, we are not really interested to consider all the things that constitute a real-world system. We call those things, in which we are interested, *entities*, including: physical objects, events and certain material relationships. This choice implies that we do not want to include more abstract concepts, like relators or qualities, in a simulation model.

All these kinds of entities are classified with the help of *entity types*, as explained in the next Subsection. Entity types allow describing the entities classified by them with the help of *attributes* and *reference properties*. Since we also want to be able to describe certain material relationships in this way, it is natural to subsume them under the concept of entities.

The Design-Time Ontology DESO-U

At design-time, we describe a discrete event system by defining the various entity types, the instances of which are part of the running system. In addition to the base concept *data type* from eUFO-0, also the concepts of (physical) object types, attributes and reference properties from eUFO-U are needed in DESO for allowing to represent abstractions of discrete event systems. In particular, the concepts of *object types* and *event types* are needed. Being special kinds of entity types, both object types and event types have *attributes* and *reference properties*, as depicted in Figure 7.

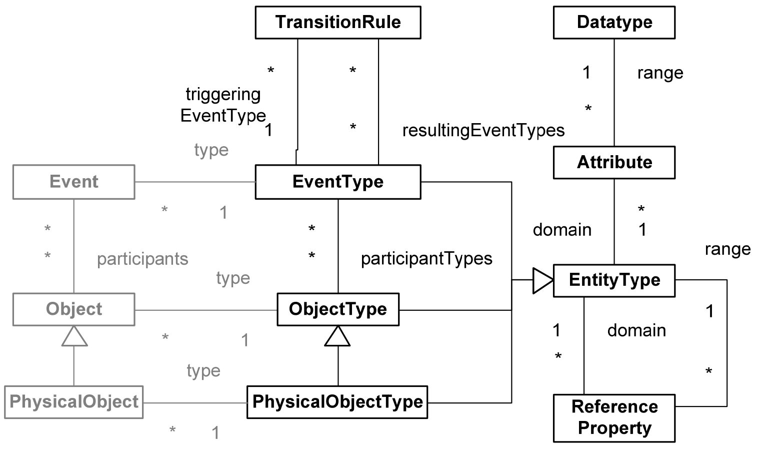

Fig. 7. The categories of types, as defined by DESO-U

4.4 ABDESO - A Foundational Ontology for ABS

We now summarize ABDESO, our *agent-based discrete event system ontology*, which extends DESO by adding the concept of *agents* and other concepts related to *agency*. Clearly, agents are special objects subsuming not only human beings, but also all kinds of living beings (including insects and bacteria), social systems (such as organizations), and also certain artificial systems (such as robots). All these objects are *interactive systems* that are able to *interact* with passive objects in their environment, or with each other, in a *purposeful* way. The question what constitutes *interaction* is closely related to the question of what is an *action*.

In philosophy, this question has been approached by asking how to distinguish "*the things that merely happen to people − the events they undergo − and the various things they genuinely do*", as [13] has put it. We define *actions* to be those events that are the direct result of the *purposeful behavior* of an *interactive system*. Notice that this definition does not exclude higher-level action concepts such as *intentional actions*, which we consider to be a special case of our more general action concept. So, we do not require an agent to have a mental state with beliefs and intentions, as it is common in 'realistic' philosophical theories of humans as cognitive subjects, and also in Artificial Intelligence approaches to multi-agent systems, in particular in the popular *Belief-Desire-Intention* (BDI) approach. Rather, we want to be able to characterize interactive systems as special objects that are distinct from passive objects, no matter if an interactive system can be considered intentional or not. It is obvious that we have to include in our account of interactive systems the concepts of *perception* and *action*. We include both of them as special kinds of events, viz *perception events* and *action events*. For being able to model communication as a special kind of interaction between agents, we introduce the concepts of a *message* and a *communication event*.

The influence of perceptions on the actions of an agent is given by its reactive behavior, which is based on behavior patterns in the form of *reaction rules*. A perception event may lead, via a reaction rule, to a resulting action of the agent in response to the event, or to an update of the agent's information state, which may include *belief*s. We assume that beliefs are expressed as *belief statements* in the agent's belief representation language. The influence of actions, and other events, on the perceptions of an agent is given by the causal laws of the agent's environment, taking the form of *transition rules*, (depicted in Fig. 7 about DESO-U), which determine the caused perception events.

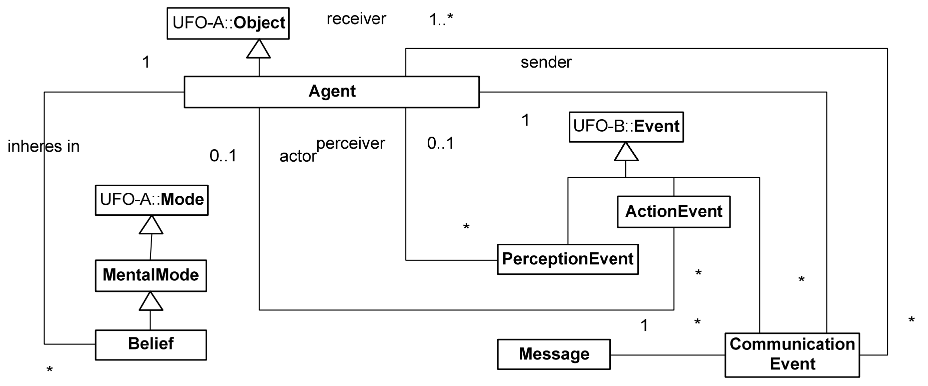

Fig. 8. The basic categories of ABDESO.

Beliefs are part of the agent's *information state*, which is the agent's basis for making action decisions. Simple belief statements take the form of entity-property-value triples, which are special cases of atomic sentences of predicate logic. These concepts are depicted in Figure 8.

A communication event is a complex event consisting of an *in-message event* (being a special case of a perception event) and an *out-message event* (being a special case of an action event). This is described in Figure 9.

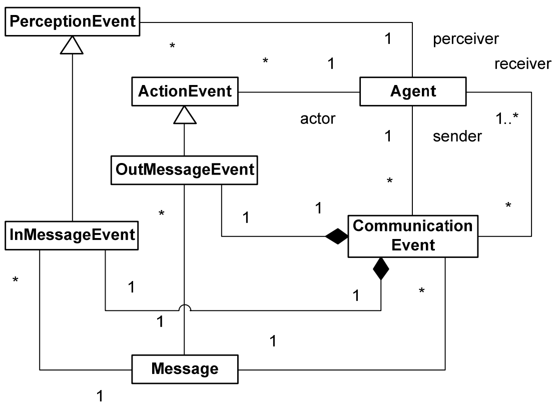

Fig. 9. In-message events and out-message events.

We claim that these seven concepts: *perception events, action events*, *messages* and *communication events*, consisting of *out-message events* and *in-message events*, as well as *beliefs*, as depicted in Figures 8 and 9, form the foundation of an ontological account of agents as interactive systems, no matter if agents are intentional or not.

In the next section, we add a few more concepts to this foundation, for being able to account for business systems, or organizations, as *institutional agents*.

Organizations as Institutional Agents

An *agent type* is associated with *reaction rules*, which define the reactive behavior of agents of that type. When a reaction rule is triggered by a perception event, this may result in zero or more action events.

Any agent has exactly one *agent base type* and, in addition, may have zero or more *agent roles* representing roles defined by an *institutional agent* (such as an organization) and played by the agent as a *subagent* of the institutional agent. The role's reactive behavior rules define implicit duties to react to certain events (resp. incoming messages).

The institutional agent's reactive behavior rules define the delegation of processing incoming messages to subagents.

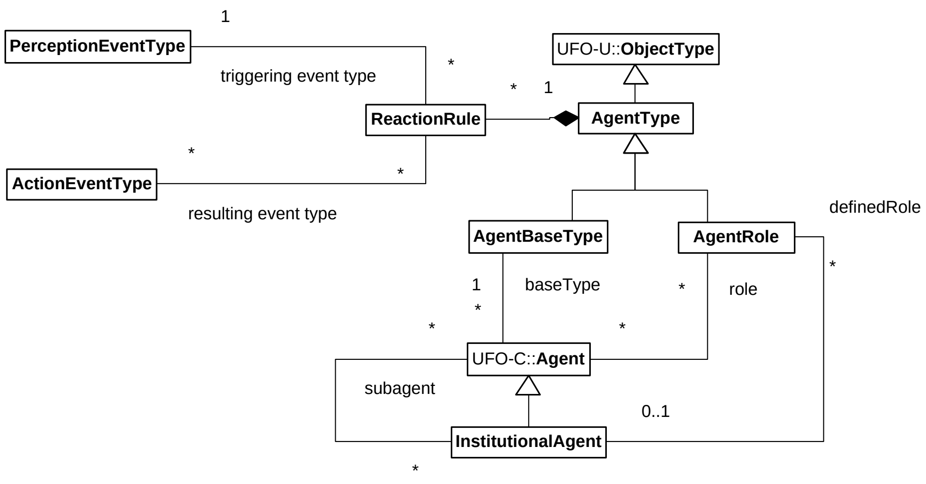

Fig. 10. Agent role types and institutional agents.

5 Soundness and Completeness of Simulation Languages

In order for an ABS model *M* to faithfully represent an agent-based discrete event system abstraction *A*, the simulation language *L* used to make *M* should faithfully represent the conceptualization of -based discrete event systems used to conceive the abstraction *A*. An ABS language can be evaluated by comparing a representation of its concepts (typically provided by a metamodel of the language) to an ontology of agent-based discrete event systems. The stronger the match between them, the easier it is to communicate and reason with models made in that language.

Since ABDESO represents a general conceptualization of agent-based discrete event systems, it can also serve as a reference ontology for evaluating the simulation languages of ABS frameworks. For any given ABS language *L*, we may consider (1) a *representation* mapping from the concepts of ABDESO to the elements (or modeling primitives) of *L* and (2) an *interpretation* mapping from the elements of *L* to the concepts of ABDESO. If these mappings are far from being isomorphisms, this may indicate deficiencies (soundness and completeness problems) of *L*.

There are four properties of an ABS language to be checked in its evaluation:

1. ***Soundness***: *L* is sound wrt ABDESO iff every element of *L* has an interpretation in terms of a domain concept from ABDESO. The ***degree of soundness*** can be measured relatively as the number of *L* elements that have an ABDESO interpretation divided by the total number of *L* elements.
2. ***Completeness***: *L* is complete wrt ABDESO iff every ABDESO concept is represented by a modeling primitive of *L*. The ***degree of completeness*** can be measured relatively as the number of ABDESO concepts that are represented by an element of *L* divided by the total number of ABDESO concepts.

3. ***Lucidity***: *L* is lucid wrt ABDESO iff every element of *L* has at most one interpretation in ABDESO. The ***degree of lucidity*** can be measured relatively as the number of *L* elements that have at most one interpretation in ABDESO divided by the total number of *L* elements.
4. ***Laconicity***: *L* is laconic wrt ABDESO iff every domain concept from ABDESO is represented by at most one element of L. The ***degree of laconicity*** can be measured relatively as the number of ABDESO concepts that are represented by at most one element of *L*.

The lower these degrees are for a given ABS language, the more problems may be expected from using a model expressed in it, e.g. by communicating incorrect information and inducing the user to make incorrect inferences about the semantics of the domain.

We now apply the described method for providing a brief evaluation of the soundness and lucidity of *BPMN*.

6 BPMN

In this section, we provide a brief summary of the essential BPMN constructs (which are described in the metamodel shown in Figure 8). BPMN defines Business Process Diagrams, which may describe processes of different types. We focus on modeling proprietary (also called "private") business processes, i.e. processes that allow a complete pre-scriptive specification of their workflow since they happen within one domain of control. Notice that what is called a *Process* in BPMN, is, in fact, rather a *process type* or *process specification*. The same terminological looseness also holds for most other BPMN terms (e.g. an *Event* is an event type).

A *Process* may be associated with a *Pool*, which is visually rendered as a rectangular container and corresponds to the business actor, or *Participant*, "owning" the process. A Pool may be compartmented into several *Lanes*, informally representing associated actors, each containing some part of the Process associated with the actor under the same domain of control. A Process essentially consist of 'flow objects' (*Events*, *Activities* and *Gateways*), 'connectivity objects' (S*equence Flows*, *Message Flows* and *Associations*) and 'artifacts' (like *Messages* and *Data Objects*).

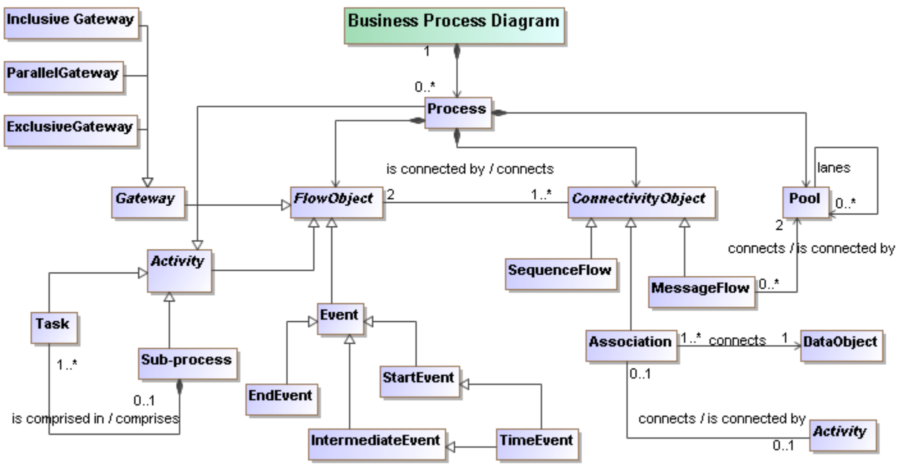

Fig. 8. BPMN core elements

Events (rendered as circles) and Activities (rendered as rectangles with rounded corners) are sequenced with the help of Sequence Flows (rendered as solid arrows) and Gateways (rendered as diamonds), which allow for AND, XOR as well as OR splits and joins of control flow branches.

Activities subsume *Tasks*, which represent atomic Activities (defined with the help of an attribute called *taskType*), and *Sub-Processes*. Tasks can be assembled into Sub-Processes, for allowing reuse. Pools are connected through Message Flows that represent message exchange patters.

Since BPMN is a process modeling language, a full simulation model may be obtained by combining an information model (e.g. in the form of a UML class model) with one or more BPMN process models.

7 Evaluation of BPMN

We evaluate BPMN by considering 1) the *interpretation* of its core elements in ABDESO, and 2) the *representation* of the ABDESO core concepts with BPMN. Since we do not consider the issue of event composition in this preliminary analysis, we exclude *Sequence Flows* and *Gateways* from the core concepts under consideration. We also exclude the abstract super-concepts *Flow Object*, *Event*, *Activity* and *Connectivity Object*, which are non-terminals in the abstract syntax tree defined by the BPMN metamodel, and, therefore, do not really count.

7.1 Evaluating Soundness and Lucidity

As a result of our preliminary evaluation of BPMN, Table 1 lists the elements of a relevant BPMN subset together with their interpretation in ABDESO.

Table 1: The ABDESO interpretation of BPMN elements

BPMN element	***ABDESO concept(s)***
Pool	Agent (Type), Physical Agent (Type)
Participant	Agent (Type), Physical Agent (Type)
Lane	Sub-Agent (Type)
Process	Complex Event Type
Task	Action Event Type
Sub-Process Activity	Complex Event Type
Start Event	Event Type
Intermediate Event	Event Type
End Event	Event Type
Message	Message Type
Message Event	In-Message Event Type, Out-Message Event Type
Message Flow	Communication Event Type
Data Object	Object Type
Soundness (Lucidity)	***100% (70%)***

As we can see from this table, the considered core subset of BPMN is 100% sound wrt ABDESO, since all elements of it have an ABDESO interpretation. It is 70% lucid with respect to ABDESO, since 4 out of 13 elements have more than one ABDESO interpretation.

7.2 Evaluating Completeness and Laconicity

When evaluating the completeness and laconicity of BPMN with the help of a representation mapping, we do not include the abstract concepts of ABDESO (Entity, Fact Triple, Entity Type, Agent Type and Mental Mode). In general, such abstract concepts (or *language elements*) correspond to non-terminals in the abstract syntax tree defined by the ontology metamodel. They serve as auxiliary super-concepts, so they do not count as much as the concrete concepts, which are terminals.

As we can see from Table 2 below, the considered core subset of BPMN is 60% complete wrt ABDESO, since 15 out of 25 ABDESO concepts can be represented in BPMN. It is 32% laconic wrt ABDESO, since 8 out of 25 ABDESO concepts have a unique representation.

The high degree of incompleteness of BPMN results mainly from BPMN's shortcomings in its information modeling capabilities. Unfortunately, the designers of BPMN decided to ignore object-oriented information modeling concepts, as defined by UML class diagrams. Therefore, BPMN does not support the concepts of attributes and reference properties for its *Data Objects*, which correspond to object types in weak sense, only.

Table 2: The representation of ABDESO concepts in BPMN

DESO concepts (11)	***BPMN (4/2)***
Object type	Data object
Physical object type	Data object
Attribute	
Datatype	
Reference property	
Atomic event type	Start Event, Intermediate Event, End Event
Complex event type	Process, Sub-Process Activity
Object destruction event type	
Object change event type	
Object creation event type	
Transition rule	
ABDESO concepts (14)	***BPMN (11/6)***
Agent (base type)	Participant, Pool
Agent role	Participant, Pool
Physical agent (base type)	Participant, Pool
Physical agent role	Participant, Pool
Institutional agent (type)	Participant, Pool
Sub-Agent (Type)	Lane
Action event type	Task
Perception event type	
Message type	Message
Communication event type	Message Flow
Out-message event type	Message Event
In-message event type	Message Event
Belief (entity) type	
Reaction rule	
Completeness (Laconicity)	***60% (32%)***

8 Conclusions

Using the *Agent-Based Discrete Event Simulation Ontology (ABDESO)*, extending the *Discrete Event Simulation Ontology (DESO),* and derived from the *Unified Foundational Ontology (UFO)*, we have evaluated the suitability of BPMN as a business simulation language. The preliminary result of 100% soundness suggests that the core elements of BPMN have been well-chosen. However, the results of only 70% lucidity, 60% completeness and 32% laconicity suggest that there are quite a few ***ambiguous*** elements, ***missing*** concepts, and ***redundant*** elements in BPMN.

Our results are preliminary, since we did not yet investigate all relevant BPMN 2 elements, but only a core part of BPMN. In future work, we plan to extend our analysis to a more complete core fragment of BPMN and also discuss each identified deficiency in some detail.

Acknowledgments. The research of the first author has been partially supported by FAPES (Grant #45444080/09) and CNPq (Grant #481906/2009-6 and Productivity Grant #309382/2008).

References

1. BPMN, The Business Process Modeling Notation, http://www.omg.org/spec/BPMN/
2. Bottazzi E., Ferrario, R.: Preliminaries to a DOLCE Ontology of Organizations, In: International Journal of Business Process Integration and Management, vol. 4, no. 4, pp. 225–238, (2009)
3. Christley, S., Xiang, X., Madey, G.: An Ontology for Agent-Based Modeling and Simulation. In Proceedings of the Agent 2004 Conference, (2004)
4. Green, P., Rosemann, M.: Business Systems Analysis with Ontologies. Idea Group Publishsing, (2005)
5. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models, PhD Thesis, University of Twente, The Netherlands, (2005)
6. Guizzardi, G., Falbo, R.A., Guizzardi, R.S.S.: Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The Case of the ODE Software Process Ontology. In XI Ibero-American Workshop on Requirements Engineering and Software Environments (IDEAS'2008), Recife, (2008)
7. Guizzardi, G., Wagner, G.: A Unified Foundational Ontology and some Applications of it in Business Modeling. In: Janis Grundspenkis and Marite Kirikova (eds.), In Proceedings of the CAiSE'04 Workshops. Faculty of Computer Science and Information Technology, Riga Technical University, Riga, Latvia, June 7–11, vol.3, pp.129–143. Riga, Latvia, (2004)
8. Guizzardi, G., Wagner, G.: Towards Ontological Foundations for Agent Modeling Concepts using UFO. In Agent-Oriented Information Systems (AOIS), selected revised papers of the Sixth International Bi-Conference Workshop on Agent-Oriented Information Systems 2005. Lecture Notes in Computer Science, vol. 3508, pp.110–124, Springer Berlin/Heidelberg, (2005)
9. Guizzardi, G., Wagner, G.: Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages. In: Poli, R. (Ed.), Theory and Application of Ontologies. Springer Berlin/Heidelberg, (2010)
10. Guizzardi, G., Wagner, G.: Towards an Ontological Foundation of Discrete Event Simulation. In B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan (eds.) In Proceedings of the 2010 Winter Simulation Conference. December 2010, Baltimore (MD), USA. pp. 652–664, (2011), http://www.informs-sim.org/wsc10papers/059.pdf
11. Livet, P., Müller, J.-P., Phan, D., Sanders, L.: Ontology, a Mediator for Agent-Based Modeling in Social Science. Journal of Artificial Societies and Social Simulation vol.13, no.1, (2010), http://jasss.soc.surrey.ac.uk/13/1/3.html
12. Wilson, G.: Action. In Edward N. Zalta (ed.), The Stanford Encyclopedia of Philosophy (Fall 2009), http://plato.stanford.edu/archives/fall2009/entries/action/
13. Guizzardi, G.,Wagner, G.: Towards an Ontological Foundation of Agent-Based Simulation. In: S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu (eds.) Proceedings of the 2011 Winter Simulation Conference, (December 2011), Phoenix, Arizona, USA